Studies in Computational Intelligence 867

Witold Pedrycz
Shyi-Ming Chen *Editors*

Development and Analysis of Deep Learning Architectures

Springer

Studies in Computational Intelligence

Volume 867

Series Editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland

The series “Studies in Computational Intelligence” (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

The books of this series are submitted to indexing to Web of Science, EI-Compendex, DBLP, SCOPUS, Google Scholar and Springerlink.

More information about this series at http://www.springer.com/series/7092

Witold Pedrycz · Shyi-Ming Chen
Editors

Development and Analysis of Deep Learning Architectures

Springer

Editors
Witold Pedrycz
Department of Electrical
and Computer Engineering
University of Alberta
Edmonton, AB, Canada

Shyi-Ming Chen
Department of Computer Science
and Information Engineering
National Taiwan University of Science
and Technology
Taipei, Taiwan

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-030-31763-8 ISBN 978-3-030-31764-5 (eBook)
https://doi.org/10.1007/978-3-030-31764-5

Preface

The paradigm of deep learning has achieved a wealth of successes, came with a plethora of concepts, methodologies, and ensuing algorithms and applications. Today, we are witnessing visible progress in this dynamically growing area. The growing interest is present in academe and industry, business, healthcare, environment science, and many others.

Ten chapters, forming this volume, are a genuine reflection of the diversity and a visible spectrum of algorithms and applications which make the underlying idea of deep learning so attractive and heavily researched nowadays. The topics covered here span a plethora of topics. Chapter "Direct Error Driven Learning for Classification in Applications Generating Big-Data" elaborates on mechanisms of learning carried out in the environment of big data; yet another timely topic quite visibly associated with deep learning. Processes of sensor design are discussed in the chapter "Deep Learning for Soft Sensor design". The application of deep convolutional networks to healthcare is covered in the chapter "Case Study: Deep Convolutional Networks in Healthcare". Domain adaptation for regression is presented in the chapter "Deep Domain Adaptation for Regression". Applications to autonomous driving, speaker recognition, baby cry detection, industrial control, wireless communication, and text analysis, the chapters "Deep Learning-Based Pedestrian Detection for Automated Driving: Achievements and Future Challenges"–"Identifying Extremism in Text Using Deep Learning", are the testimony of a wealth of usages of deep learning.

We would like to express our thanks to Prof. Janusz Kacprzyk, the Series Editor-in-Chief, for his ongoing encouragement and support when realizing this publishing project. We are indebted to the professionals at Springer; the team has made the overall production process smooth and efficient.

Edmonton, Canada — Witold Pedrycz
Taipei, Taiwan — Shyi-Ming Chen

Contents

Direct Error Driven Learning for Classification in Applications Generating Big-Data 1
R. Krishnan, S. Jagannathan and V. A. Samaranayake
1 Introduction 2
2 Background and Preliminaries 4
  2.1 Related Work 7
3 Nonlinear Dimensionality Reduction 9
  3.1 Stage 1: Groupings at the First Step $i = 1$ 11
  3.2 Stage 2: The Transformation 11
  3.3 Stage-1: Groupings at the Later Steps that is $i > 1$ 13
4 Classification 14
5 Two-Step Classification Framework 18
6 Results and Discussion 20
  6.1 Mitigating Noisy Dimension 21
  6.2 Mitigating Heterogeneity 22
  6.3 Classification 24
7 Conclusions and Future Work 27
References 27

Deep Learning for Soft Sensor Design 31
Salvatore Graziani and Maria Gabriella Xibilia
1 Soft Sensors in the Process Industry 31
2 State of the Art of SSs Based on Deep Learning 33
3 SSs Design 36
4 Deep Belief Networks 38
5 Real-World Cases Study 40
  5.1 Semi-supervised Learning Based Soft Sensors 40
  5.2 Reducing Model Complexity by Using DBN 43
  5.3 Gaining Insight on the SS Structure by Exploiting DBN Features 51

6 Conclusion 56
References 57

Case Study: Deep Convolutional Networks in Healthcare 61
Mutlu Avci, Mehmet Sarıgül and Buse Melis Ozyildirim
1 Introduction 62
2 Deep Learning Methods in Healthcare 64
2.1 Convolutional Neural Networks 64
2.2 Autoencoders 65
2.3 Deep Polynomial Network 67
2.4 Hybrid Methods 67
2.5 Convolutional Neural Network Based Methods 69
2.6 Techniques Used in Deep Learning Methods 73
3 Deep Learning Applications in Healthcare 77
3.1 Breast Mass and Microcalcifications Detection for Breast Cancer Classification 77
3.2 Brain Tumor Segmentation and Alzheimer's Disease 81
3.3 Diabetic Retinopathy and Cataract 84
3.4 Cardiology 85
3.5 Lung Cancer Detection 86
4 Conclusion and Future Works 87
References 87

Deep Domain Adaptation for Regression 91
Ankita Singh and Shayok Chakraborty
1 Introduction 91
2 Related Work 94
3 Proposed Framework 96
3.1 Supervised Loss on Labeled Data 97
3.2 MMD Loss Between Source and Target Data 97
3.3 Semi-supervised Loss for Unlabeled Target Data 98
4 Experiments and Results 99
4.1 Datasets and Feature Extraction 99
4.2 Comparison Baselines 100
4.3 Deep Network Architecture 101
4.4 Experimental Setup 102
4.5 Performance Metrics 103
4.6 Performance Analysis 103
4.7 Effect of Labeled Target Data 106
4.8 Effect of Labeled Source Data 107
4.9 Effect of Kernel Function 108
4.10 Feature Analysis 109
5 Conclusion and Future Work 110
References 111

Deep Learning-Based Pedestrian Detection for Automated Driving: Achievements and Future Challenges 117
Michelle Karg and Christian Scharfenberger
1 Introduction 118
2 Safety Relevance of Pedestrian Detection for Advanced Driver Assistance Systems and Automated Driving 120
3 Pedestrian Detection 123
3.1 Convolutional Neural Networks for Pedestrian Detection 124
3.2 The Dependence of the Error Rate on Task Complexity 129
3.3 Overview on Public Research Datasets for Pedestrian Detection 132
3.4 Evaluation Metrics for Pedestrian Detection 133
3.5 Discussion and Future Directions 135
4 Summary and Outlook 137
References 140

Deep Learning in Speaker Recognition 145
Omid Ghahabi, Pooyan Safari and Javier Hernando
1 Introduction 145
2 Conventional Techniques 148
2.1 Feature Extraction 149
2.2 Supervectors and i-Vectors 150
2.3 i-Vector Scoring 151
3 Deep Learning Frontends 152
3.1 Feature and i-Vector Extraction 152
3.2 Supervised Speaker Embeddings 155
3.3 Unsupervised Speaker Embeddings 157
4 Deep Learning Backends 160
5 Deep Learning End-to-Ends 162
6 Conclusions 164
References 165

Baby Cry Detection: Deep Learning and Classical Approaches 171
Rami Cohen, Dima Ruinskiy, Janis Zickfeld, Hans IJzerman and Yizhar Lavner
1 Introduction 172
1.1 Approaches in Audio Event Detection 173
1.2 The Origin and Role of Infant Cry 174
2 Deep Learning Approach 176
2.1 Data Representation 176
2.2 Feed-Forward Architectures 178
2.3 Recurrent Neural Networks (RNNs) 181
3 Classical Approaches 182
3.1 Preprocessing and Feature Extraction 182

3.2 Logistic Regression 186
3.3 Support Vector Machine 188
4 Performance Evaluation 189
4.1 Database 189
4.2 Training and Test Process 189
4.3 Results 189
5 Conclusion 192
References 193

Securing Industrial Control Systems from False Data Injection Attacks with Convolutional Neural Networks 197
Sasanka Potluri, Shamim Ahmed and Christian Diedrich
1 Introduction 198
1.1 Network Attacks 201
1.2 Injection Attacks 202
2 State of the Art 202
3 Proposed Approach 204
3.1 Pre-processing 204
3.2 Feature Extraction 205
3.3 Deep Learning 206
4 Dataset 208
5 Implementation 210
5.1 Pre-Processing 210
5.2 Image Representation 214
5.3 Training of CNN 215
6 Results and Discussion 217
7 Conclusion and Future Works 219
References 220

Deep Learning for Wireless Communications 223
Tugba Erpek, Timothy J. O'Shea, Yalin E. Sagduyu, Yi Shi and T. Charles Clancy
1 Introduction 224
2 Deep Learning for End-to-End Communication Chain 226
2.1 Single Antenna Systems 228
2.2 Multiple Antenna Systems 231
2.3 Multiple User Systems 236
3 Deep Learning for Spectrum Situation Awareness 238
3.1 Channel Modeling and Estimation 239
3.2 Signal Detection and Modulation Classification 245
3.3 Generative Adversarial Methods for Situation Awareness 249
4 Deep Learning for Wireless Communications Security 251
4.1 Operational Modes for Transmitter and Adversary 253
4.2 Jamming Based on Exploratory Attack 255

4.3 Other Attacks Based on Adversarial Deep Learning 259
4.4 Defense Against Adversarial Deep Learning 260
5 Conclusion 262
References 263

Identifying Extremism in Text Using Deep Learning 267
Andrew Johnston and Angjelo Marku
1 Introduction 267
2 Background 271
3 Dataset 276
3.1 Sunni Extremism 277
3.2 White Nationalism 278
3.3 Antifa 279
3.4 Sovereign Citizens 280
3.5 Benign Material 281
4 Building the Models 283
5 Results and Analysis 283
6 Related Work 285
7 Future Work 286
8 Conclusion 287
References 287

Index 291

Direct Error Driven Learning for Classification in Applications Generating Big-Data

R. Krishnan, S. Jagannathan and V. A. Samaranayake

Abstract In this chapter, a comprehensive methodology is presented to address important data-driven challenges within the context of classification. First, it is demonstrated that challenges, such as heterogeneity and noise observed with big/large data-sets, affect the efficiency of a deep neural network (DNN)-based classifiers. To obviate these issues, a two-step classification framework is introduced where unwanted attributes (variables) are systematically removed through a preprocessing step and a DNN-based classifier is introduced to address heterogeneity in the learning process. Specifically, a multi-stage nonlinear dimensionality reduction (NDR) approach is described in this chapter to remove unwanted variables and a novel optimization framework is presented to address heterogeneity. In NDR, the dimensions are first divided into groups (grouping stage) and redundancies are then systematically removed in each group (transformation stage). The two-stage NDR procedure is repeated until a user-defined criterion controlling information loss is satisfied. The reduced dimensional data is finally used for classification with a DNN-based framework where direct error-driven learning regime is introduced. Within this framework, an approximation of generalization error is obtained by generating additional samples from the data. An overall error, which consists of learning and approximation of generalization error, is determined and utilized to derive a performance measure for each layer in the DNN. A novel layer-wise weight-tuning law is finally obtained through the gradient of this layer-wise performance measure where the overall error

This research was supported in part by an NSF I/UCRC award IIP 1134721 and Intelligent Systems Center.

R. Krishnan (✉) · S. Jagannathan
Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO, USA
e-mail: krm9c@mst.edu

S. Jagannathan
e-mail: sarangap@mst.edu

V. A. Samaranayake
Department of Mathematics and Statistics, Missouri University of Science and Technology, Rolla, MO, USA
e-mail: vsam@mst.edu

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_1

is directly utilized for learning. The efficiency of this two-step classification approach is demonstrated using various data-sets.

Keywords Deep learning · Big-data · Dimensionality reduction · Learning regime

1 Introduction

The explosion of digital data has highlighted the need for sustainable analysis methods in many applications [9] such as fault diagnostics, object recognition and others. For instance, in the problem of classification, many big/large data-sets are commonly found and need to be analyzed. These data-sets are characterized by large (n, p), where n refers to the total number of data-points while p denotes the total number of dimensions in the data set. To obtain reasonable efficiency while analyzing such data-sets several associated challenges must be addressed as Big-data-sets are distinguished by high dimensionality where redundant dimensions are observed [6, 12].

When high dimensionality is mixed with massive sample sizes, there is a need for distributed storage [51]. As a result, one must rely upon data-samples for sustainable analysis. In the absence of an intelligent approach to aggregate information across these data-samples, several complications, including experimental variation and statistical bias can be observed [9]. In addition, with an increase in the data-dimensions, the number of data points available at any one storage unit becomes smaller [51]. Due to this, one may observe irregular or incorrect estimation of important statistical parameters thus increasing error [9] which refers to the challenge of heterogeneity.

A few of these challenges can be addressed using a deep neural network (DNN), where the parameters of a parametric map are estimated by minimizing a performance measure defined according to the problem at hand. However, as the number of parameters in a parametric map scales very quickly with data dimensions [9], there are estimation errors. To obviate these issue, dimension-reduction procedures are commonly utilized.

Popular dimension-reduction approaches including principal component analysis (PCA) [55], factor analysis [22] and others [22] rely on correlation or covariance matrices. However, it is not possible to estimate dependencies efficiently [22] because correlation matrices are either low rank or computationally expensive when high dimensional data-sets are considered. Overall, traditional dimension-reduction approaches [22, 55] are inefficient in large dimensional scenarios due to a one-step mapping common with these approaches.

On the other hand, heterogeneity and noise can [9] adversely impact DNN learning where an increase in generalization error is observed. In addition, a common way of learning parameters in a DNN-based approach is stochastic gradient descent (SGD) [29]. As SGD-based learning [29] can suffer from unstable learning when the number of layers in the DNN is very large [36], the traditional learning process is also inappropriate for big-data classification.

To alleviate these issues, this chapter addresses the impact of big-data challenges by (1) reducing the generalization error due to heterogeneity and data-noise via a novel classification framework, (2) improving the learning efficiency in the presence of unwanted dimensions by using a dimension reduction approach prior to classification, and (3) mitigating vanishing gradients observed with traditional SGD by introducing a novel learning scheme.

A two-step process is introduced in this chapter. The first step in this procedure is nonlinear dimension-reduction (NDR) where the issue of unwanted dimensions is addressed while incorporating nonlinear relationships among dimensions. In NDR, the p attributes are first organized into groups. In each group, dependencies among the attributes are measured through distance covariance [47], that captures nonlinear dependencies among attributes. Next, a singular value decomposition (SVD)-based low-dimensional approximation is estimated for each group. Once every group is transformed, the attributes are reorganized to form groups and transformed again. During the transformation process, the group-wise organization is performed by utilizing the magnitude of singular values. A singular value-based user condition is introduced to determine the number of dimensions that must be extracted from the data. The criterion also acts as a stopping condition for the dimension reduction process.

Using the data in the reduced dimensions, a DNN-based classifier is employed in the second step of classification. The classifier is designed in such a way that an approximate generalization error is minimized as part of the optimization process. The approximation of generalization error is obtained by introducing synthetic distortions in a neighborhood around a data-sample. A learning problem is then designed to estimate the parameters of the classifier to minimize the overall cost comprised of learning and approximate generalization errors.

To mitigate the issue of vanishing gradients, a direct error-driven learning (EDL) regime is introduced where the weights at each layer of the DNN learn directly through the overall error. To enable this learning, a cost function for each layer in the DNN is defined as a function of the overall cost. The optimization procedure is designed in such a way that the learning at each layer is independent of other layers in the DNN. Finally, to update the parameters of the dimension-reducing transformation, novel batch-wise updates are introduced.

The two-step classification framework presented in this chapter can be thought of a combination of two deep architectures where the first block removes unwanted dimensions and the second block acts as a classifier to the data-points. In addition, both these block address important challenges that arise while analyzing big-data and support each other to provide efficient classification. The efficiency of the overall approach is demonstrated using many big-data sets. In the next section, the notations and other preliminaries are discussed.

2 Background and Preliminaries

In this section, the mathematical background is first described and the problem under consideration is formulated and a summary of all the notations is provided in Table 1. To start, consider any data generating process where $\boldsymbol{x} \in \mathbb{R}^{n \times p}$ denotes a data sample. Consider now the objective of determining whether the system is at healthy state or at a fault through a map $\phi(.)$. The purpose of the map is to transform the data-sample into probability domain where the magnitude of probabilities indicates category/class corresponding to a data-point. The overall learning process can then be defined as that of estimating this map through a parametric form given a training data-set $\boldsymbol{\mathcal{X}}$ and its labels $\boldsymbol{\mathcal{Y}}$. Formally, it is of importance to evaluate

$$p(\boldsymbol{x} \in \boldsymbol{\Psi}_m | \boldsymbol{\mathcal{X}}), \tag{1}$$

where $\boldsymbol{\Psi}_m$ with $m = 1, 2, 3, \cdots, \mathcal{F}$ denotes the populations for the corresponding classes. Any typical parametric map can be used to evaluate these probabilities and examples include neural networks, spline function and others [4].

In this chapter, DNN is discussed as a parametric map for evaluating these probabilities. An example of such a classification problem can be seen in Fig. 1. Formally, one may first assume that $\phi(.)$ can be approximated through a DNN with appropriate

Table 1 Notations

Notation	Meaning
$\boldsymbol{\mathcal{X}}$	Data-set
$\mathcal{F}$	The total categories
(n, p)	Total number of sample points, total number of dimensions
i, m, l	Index for different levels in the approach with m, l defining indexes for the paper
$\boldsymbol{A}, \boldsymbol{B}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}$	Distance matrices and double-centered matrices
$\boldsymbol{C}$	Distance correlation matrix
λ	Singular-values
M	The total no. of batches in the data
j	Index for the attributes
$t: \quad 1, \cdots, T^{(i)}$	Group index
$\kappa_t^{(i)}, \eta_t^{(i)}$	Estimated number of dimensions to be extracted and the total number of dimensions
α	The variation that must be captured in a group
$(.)^{(i)}$	Number of layers in the DNN
$\boldsymbol{W}, \hat{\boldsymbol{W}}$	Ideal and estimated weights
k	Iteration index in the learning phase
ε	The approximation error
$f^{(l)}, for \quad l = 1 \cdots d$	Layer-wise activation functions with d layers

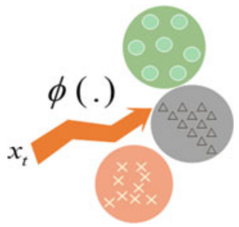


Fig. 1 Illustration for the classification problem

capacity. Let the weights be denoted as $\hat{\boldsymbol{\theta}} = [\hat{\boldsymbol{W}}^{(1)} \cdots \hat{\boldsymbol{W}}^{(d)}]$ where layer-wise bias is included in the weights. Next, assume that an ideal set of weights exists for this approximation and write $\hat{\boldsymbol{y}}(\boldsymbol{x}) = \hat{\boldsymbol{y}}(\boldsymbol{x};\theta) + \varepsilon$, implying

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = f^{(d)}(\hat{\boldsymbol{W}}^{(d)} \cdots (f^{(1)}(\hat{\boldsymbol{W}}^{(1)}(\boldsymbol{x})))) + \varepsilon. \tag{2}$$

Let $f^{(l)}$, for $l = 1 \cdots d$ denote the layer-wise activation functions with d denoting the total number of layers in the DNN. The term ε represents the approximation error which is a function of the capacity of the DNN and can be assumed constant as shown in [34]. The following assumptions are necessary for solving the learning problem.

1. The samples are obtained in such a way that they are independently and identically distributed.
2. Data-distribution in the training phase is similar to the test phase.

With these aforementioned assumptions being true, learning the weights of the DNN involves minimizing a cost function $J_o(.)$ [29] defined as a function of the expected value of the difference between ideal output(labels) and the DNN output.

Remark 1 In traditional machine learning setup, the feature extraction map and the classifier map is designed separately. In such a design, one could look at the centroid of the data-points belonging to each class to determine classes. In a DNN-based setup, these centroids are decided prior to the learning procedure and the optimization is designed to force the DNN output to become as close as possible to these prefixed centroids. Usually, the number of outputs in DNN are equal to the number of classes and each axis in the output space of the centroid is chosen to be a centroid to the class.

For this discussion, assume $k \in [1, N]$ be the iteration index for the learning phase where N is the total number of iterations. Let the difference between the labels and the DNN output be denoted as $\boldsymbol{e}_l$ (learning error) such that

$$E[\boldsymbol{e}_l(k)] = E_{\forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}, \boldsymbol{y} \in \boldsymbol{\mathcal{Y}}}[\boldsymbol{y} - \hat{\boldsymbol{y}}], \tag{3}$$

where $E[.]$ is the expectation operator and $\boldsymbol{y} \in \mathbb{R}^{\mathcal{F} \times n}$ represents the ideal output (labels) for $\phi(.)$. Let the total error that includes the learning error and the approximation error be given as

$$E[\boldsymbol{e}](k) = E[\boldsymbol{e}_l](k) + \varepsilon. \tag{4}$$

The overall cost can be obtained by squaring the total error resulting in

$$J_o = E[\boldsymbol{e}]^T E[\boldsymbol{e}] = E[\boldsymbol{e}_l]^T E[\boldsymbol{e}_l] + 2\varepsilon^T E[\boldsymbol{e}_l] + \varepsilon^T \varepsilon. \tag{5}$$

The notation for iterations in Eq. (5) is suppressed from hereon.

Remark 2 In this chapter, the discussion follows the use of quadratic cost, however, one may switch to the cross-entropy cost in practical situations.

Note that the first term in the overall cost function is the empirical error obtained through the data sample. The second term is the projection of the empirical error on the approximation capacity of the DNN and this term can be understood as the cost of generalization. The last term is basically the cost due to the approximation capacity of the DNN. With these observations, one can rewrite the overall cost by denoting the empirical cost as $J_{emp} = E[\boldsymbol{e}_l]^T E[\boldsymbol{e}_l]$, generalization cost as $J_{gen} = 2\varepsilon^T E[\boldsymbol{e}_l]$ and $J_{apx} = \varepsilon^T \varepsilon$ being the approximation cost.

$$J_o = \quad J_{emp} + J_{gen} + J_{apx}. \tag{6}$$

Typically in practical scenarios, J_{emp} is the only measured quantity in the learning phase and the unknown quantities that is J_{gen} and J_{apx} are usually assumed small and bounded. Within these constraints J_{emp} is minimized during learning [4]. As a result, a major research problem in the literature is to design the DNN model in such a way that J_{gen} is kept as small as possible. However, in the presence of big-data, the learning process does not remains standard and J_{gen} can move out of bounds. One common scenario, where this will happen is the case when the assumption of training and test distribution being similar is violated. Usually, the sampling process has to ensure that the training data-set well represents the data-distribution. However, the constraints of computing would not usually allow a sample that can densely represent a large dimensional space. The resulting problem can be observed with an increase in generalization error because the training set and the testing set are not similar anymore.

To further elaborate the issue, consider the analysis of MNIST data-set with a three-layer neural network [29]. For illustration, redundant dimensions are synthetically introduced into the data and illustrate the results in Fig. 2b. One may observe that there is a reduction in accuracy with an increase in unwanted dimensions in the data. This increase in error is generally reflected in an increased J_{emp}.

Next, for illustration of heterogeneity, consider big-data to be collected from an attribute and let it be stored at multiple locations where $\boldsymbol{\mathcal{X}}$ with mean μ_0 can be sampled from any of the available locations. One can observe that $\boldsymbol{\mathcal{X}}$ does not explain all the characteristics of Ψ as observed in Fig. 2a. Consequently, methodologies that learn from $\boldsymbol{\mathcal{X}}$ does not capture the overall distribution Ψ fully. Due to this, erroneous predictions are observed and this is reflected in an increased J_{gen}.

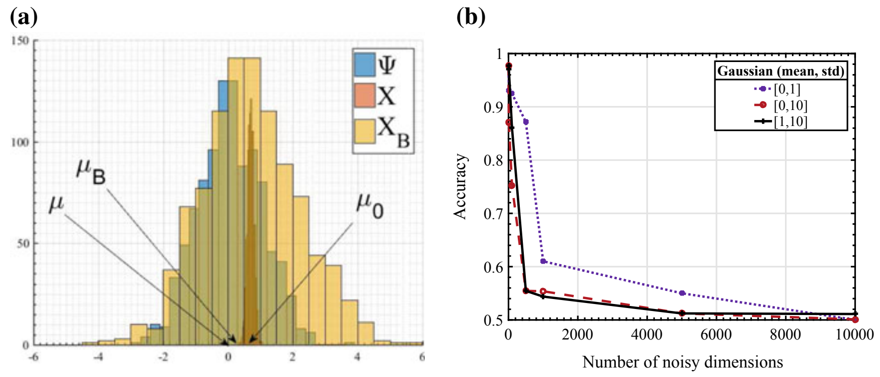


Fig. 2 **a** Histograms of ψ, $\mathcal{X}$ with $\mathcal{X}_B$ **b** Accuracy corresponding to the number of unwanted dimensions

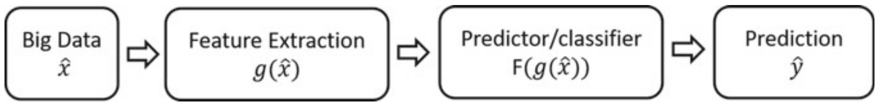


Fig. 3 Two step analysis process

In summary, the challenges introduced by the data are reflected in J_{gen} and J_{emp}. With these observations, it has been demonstrated that the presence of unwanted dimensions in the data introduces errors in classification. The main contribution of this chapter is to develop ways to address these challenges effectively. To address these challenges, a two step framework as illustrated in Fig. 3, is utilized.

These challenges have been independently addressed in the literature but not in a satisfactory manner. The related literature is discussed next.

2.1 Related Work

A most common way of addressing the challenge of noisy dimension is through feature extraction approaches. Several dimension-reduction techniques such as Isomap [2], LLE [39], Hessian LLE [8], Laplacian eigen-maps [3] and its variants [10] including kernel PCA [42], have used to this end. These methodologies discover the intrinsic geometric structure of high-dimensional data. In [12], the authors show that high dimensional spaces are sparse and suffer from distance concentration [12] which introduces complications while capturing the geometric structure.

To address the problem of computational disadvantages and imperfect estimation in large dimensional scenarios, divide-and-conquer mechanisms for dimension reduction were proposed in [1, 17, 49, 56], where the authors have shown that in many cases, known application-specific relationships and natural groupings can be exploited for efficient dimension reduction. However, such approaches [1, 17, 49, 56], require prior knowledge of these relationships to enable organization, which may not be available in Big-datascenarios.

In traditional dimension reduction approaches [1–3, 7, 8, 10, 11, 17, 21, 23, 39, 42, 49, 56], one-step mapping is performed to reduce dimensions. Since this approach is computationally unfriendly when the dimension of the original data is very large,it is not feasible in Big-datascenarios.

In contrast, a multi-step dimension-reduction approach is developed in this chapter where the dimension reduction process process does not involve a one step mapping. While standard dimension-reduction approaches require the practitioner to specify the number of dimensions to be extracted from the data. In contrast, the user only specifies the percentage of information to be retained in NDR and NDR can estimate the number of dimensions to be extracted from the data. In the dimension-reduction approaches mentioned earlier [1–3, 7, 8, 10, 11, 17, 21, 23, 39, 42, 49, 56], new information cannot be incorporated during analysis. Therefore, [1–3, 7, 8, 10, 11, 17, 21, 23, 39, 42, 49, 56], one must re-update the dimension reducing transformation or ignore the new information [11, 21, 23]. Both of these scenarios are inefficient. In contrast, batch-wise updates re introduced in this paper.

To address the challenge of heterogeneity common with classification regimes, cross-validation methods [46], norm regularization [33] and dropouts [45] are common. However, these methods [33, 45] are heuristic and do not guarantee performance.On the other hand, hyper-parameters could be selected based on upper-bounds derived through rigorous mathematical analysis using Vapnik-Chervonenki's dimensions, Radmacher complexity or uniform stability [19, 53]. However, this process is often impractical in Big-datacases. In brief, the approaches present in the literature [16, 45, 46, 53] are either heuristic and do not guarantee low generalization error or require complex mathematical analysis. By minimizing generalization error in the learning procedure, the impact of heterogeneity and data-noise is explicitly mitigated in contrast with [14, 16, 38, 45, 54].

A common learning regime used to optimize the weights of a DNN is SGD which suffers from the issue of unstable learning signals [13, 36]. Gradient scaling through ReLU activation function [36, 41], adaptive learning rates [25] and L_1/L_2 regularization [15] are common ways of addressing this issue. However, these techniques are only work-around to the real issue of unstable gradients.

To overcome the issue, target propagation was introduced in [30] where auto-encoders are used to estimate targets at every layer. However, it is difficult in most cases to define targets at each layer and the computational overload of maintaining auto-encoders at every layer is large. To alleviate this issue, direct feedback alignment was proposed in [35] where the error is directly used for learning at every layer. Although impressive results have been reported in [35], several drawbacks exist. First, the approach allows the learning signal to progress in random directions. Second, zero-derivative activation functions would impede learning.

In contrast with traditional gradient descent [19, 33, 36], the feedback is designed to ensure that the magnitude of the learning signal does not vanish while the update rule is generic enough to solve any cost function. In contrast with [35], the user-defined matrix is specifically designed to propel learning in the directions of steepest descent and the learning is not inhibited even when the activation function is saturated.

As a solution to this issue, a two step approach is introduced in this chapter where dimensions in the data are reduced prior to classification. A pictorial representation is given in Fig. 3. Recently, several researchers [20, 24, 44, 55] have introduced techniques based on two-step approach for fault diagnostics and classification. In these efforts, popular dimension reduction methods such as principal component analysis (PCA) [22, 55], factor analysis are frequently used for reducing dimensions and a standard classification setup is utilized for final detection.

However, these methods [20, 24, 44, 55] rely on correlation or covariance matrices for dimension reduction. Due to the challenges posed by high dimensional spaces discussed before, it is not possible to estimate these matrices correctly. Moreover, the process is computationally in-feasible.

To address the problem of computational disadvantages and imperfect estimation in large dimensional scenarios, divide-and-conquer mechanisms for dimension reduction were proposed in [1, 17, 49, 56], where the authors have shown that in many cases, known application-specific relationships and natural groupings can be exploited for efficient dimension reduction. However, such approaches [1, 17, 49, 56], require prior knowledge of these relationships to enable organization, which may not be available in big-data scenarios. In the next few sections, these challenges are addressed independently. The challenge due to unwanted dimensions is discussed first.

3 Nonlinear Dimensionality Reduction

It has been demonstrated that dimension reduction is inherently inefficient because they depend on a measure of dependence that does not work well when nonlinear relationships are presents or when the number is dimensions is very high.

Since independence of random variable cannot be reliably determined while using Pearson correlation, there is a need for a novel dependence measure. One option is the use of distance covariance (DC) that was introduced in [47]. For the purpose of completeness, a brief description of this dependence measure is provided next. However, for additional details on distance covariance, please refer to [47] and the references therein.

Let there be only two attributes in the data that is ($p = 2$) and let a vector from $\boldsymbol{\mathcal{X}}$ be described as $[\boldsymbol{a} \quad \boldsymbol{b}]$. Consider the problem of determining the strength of relationship between $\boldsymbol{a}$ and $\boldsymbol{b}$. To this end, define $\boldsymbol{a} = [a_1 \quad \cdots \quad a_n]^T$ and $\boldsymbol{b} = [b_1 \quad \cdots \quad b_n]^T$. Next, determine the pairwise distances between elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ such that they are denoted by $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively. Note that $\boldsymbol{A} = \{\boldsymbol{A}_{m,l} : m, l = 1, 2, \cdots n\}$ is the set of pairwise distances between all elements in $\boldsymbol{a}$ and $\boldsymbol{B} = \{\boldsymbol{B}_{m,l} : m, l = 1, 2, \cdots n\}$ is between elements of $\boldsymbol{b}$. Considering euclidean distances to evaluate distances, for any two elements m and l, one can write $\boldsymbol{A}_{m,l} = (a_m - a_l)^2$ and $\boldsymbol{B}_{m,l} = (b_m - b_l)^2$.

Let $(.)_{m.}$ describe the m^{th} row in (.) along with $(.)_{.l}$ describing the l^{th} column in (.). Lets double center the pairwise distance matrices to get

$$\tilde{\boldsymbol{A}}_{m,l} = \begin{cases} \boldsymbol{A}_{m,l} - \frac{1}{n}\mathbf{1}_n^T(\boldsymbol{A})_{.l} - \\ \frac{1}{n}(\boldsymbol{A})_{m.}\mathbf{1}_n + \frac{1}{n^2}\mathbf{1}_n^T\boldsymbol{A}\mathbf{1}_n \\ 0 \end{cases} \quad \tilde{\boldsymbol{B}}_{m,l} = \begin{cases} \boldsymbol{B}_{m,l} - \frac{1}{n}\mathbf{1}_n^T(\boldsymbol{B})_{.l} - \\ \frac{1}{n}(\boldsymbol{B})_{m.}\mathbf{1}_n + \frac{1}{n^2}\mathbf{1}_n^T\boldsymbol{B}\mathbf{1}_n & m \neq l \\ 0 & m == l, \end{cases} \tag{7}$$

where the column vector is denoted as $\mathbf{1}_n$ with length n. With these notation, the sample distance covariance/correlation can now be defined.

Definition 1 Given finiteness of second order moments for $\boldsymbol{a}$ and $\boldsymbol{b}$, one can define a sample estimate of distance covariance $\nu_n^{\boldsymbol{a},\boldsymbol{b}}$ as

$$\nu_n^{\boldsymbol{a},\boldsymbol{b}} = \frac{1}{n^2}\mathbf{1}_n^T(\tilde{\boldsymbol{A}} \odot \tilde{\boldsymbol{B}})\mathbf{1}_n, \tag{8}$$

where the Hadamard product is denoted by $\odot$ and the vector of ones is denoted by $\mathbf{1}_n$ with size n. One may write the squared sample distance correlation $r_n^{\boldsymbol{a},\boldsymbol{b}}$ as

$$r_n^{\boldsymbol{a},\boldsymbol{b}} = \frac{\nu_n^{\boldsymbol{a},\boldsymbol{b}}}{\sqrt{\nu_n^{\boldsymbol{a},\boldsymbol{a}}\nu_n^{\boldsymbol{b},\boldsymbol{b}}}}. \tag{9}$$

Using distance covariance/correlation as a measure of dependence among any two dimensions, a dimension reduction process is developed next that is effective in large dimensional cases and does not suffer from the rank deficiency problem.

The basic idea behind this approach is to group the dimensions together such that the size of the group is smaller than the number of data-points and then transform these groups to reduce dimensions. This grouping and transformation process is to be continued for multiple steps or till the stopping criterion is satisfied. Thus, the overall methodology can be divided into I steps where each step is composed of two stages, the grouping stage and the transformation stage. The notation indicating the step index is i and the value of I is determined based on the number of dimensions that must be extracted from the data or the stopping criterion. Before presenting NDR, some preliminary notations are established.

Let there be M batches in the data where $n = n_1 + n_2 + \cdots + n_M$ and let b_q represents a batch of data with q as index. It follows that the data-set is composed of M batches and $\boldsymbol{\mathcal{X}} = \{\cup_{l=1}^{M}\boldsymbol{\mathcal{X}}_l\}$. Let $\boldsymbol{\mathcal{X}}_1$ denote a batch of data and consider the data to consists of eight dimensions such that $\boldsymbol{\mathcal{X}}_1 = \boldsymbol{X}^{(1)} = [\boldsymbol{x}_1^{(1)} \quad \cdots \quad \boldsymbol{x}_8^{(1)}]$ where $\boldsymbol{x}_j^{(1)}$ represents the j^{th} column in $\boldsymbol{X}^{(1)}$ and $\boldsymbol{x}_j^{(1)}$ denotes a vector of size n_1. The superscript in the symbols defined before denote the step in the dimension reduction procedure. First, the overall approach(both NDR and the DNN) is discussed using one batch of data and then in the end information is aggregated across multiple batches. Each step in NDR can be described as a composition of two stages and discussion begins with the grouping stage at the first step.

3.1 Stage 1: Groupings at the First Step $i = 1$

At the first step, there is no information about any relationships in the data, therefore, the dimensions are grouped at random. First, group the eight dimensions in the example as $(1, 2), (3, 4), (5, 6), (7, 8)$ where $(\boldsymbol{x}_1^{(1)}, \boldsymbol{x}_2^{(1)})$ forms the first group with $t = 1$ and $(\boldsymbol{x}_3^{(1)}, \boldsymbol{x}_4^{(1)})$ forms the second group with $t = 2$ and so on. If any predefined relationships exist, the practitioner may define these groups accordingly. Let the data-batch at each *ith* step within the *tth* group be denoted as $\boldsymbol{X}_t^{(i)}$ and initiate the transformation stage.

3.2 Stage 2: The Transformation

The basic idea of transformation is to evaluate the distance covariance (DC) among all dimensions in every group and reduce dimensions through the SVD of distance covariance. To enable this, let $\boldsymbol{C}_i^{(t)}$ denotes the DC matrix at any step i and the group t. For the first step in our illustrative example $i = 1$ and $t = 1, 2$. Using the DC matrix, the data is transformed in such a way that redundant attributes are minimized and the variance is maximized [23]. In other words, important dimensions (determined by their respective variance) must remain at the end of the transformation. This can be achieved by retaining the dimensions corresponding to the large singular-values. Thus, define the SVD of $\boldsymbol{C}_t^{(i)}$ as

$$\boldsymbol{C}_t^{(i)} = \boldsymbol{U}_t^{(i)} \boldsymbol{\Sigma}_t^{(i)} \boldsymbol{V}_t^{(i)^T}. \tag{10}$$

Note that the singular-values $\boldsymbol{C}_t^{(i)}$ can be written as $\lambda_{t,1} \geq \lambda_{t,2} \geq \cdots \geq \lambda_{t,\eta_t^{(i)}} \geq 0$ and $\boldsymbol{\Sigma}_t^{(i)} = diag(\sqrt{\lambda_{t,1}}, \sqrt{\lambda_{t,2}}, \cdots \sqrt{\lambda_{t,\eta_t^{(i)}}})$ with $\boldsymbol{U}^{(i)}$ and $\boldsymbol{V}^{(i)}$ representing an orthonormal transformations (singular-basis).

Once, the SVD for a group is known, one may project the data in the group onto a low dimensional space. However, to determine the size of the projected space, a user-defined threshold denoted as $\alpha/100$ is utilized. Here α denotes the total information that must be retained from a group in the data. To enable this, first normalize the singular-values such that they sum to one and retain the dimensions corresponding to the largest $\kappa_t^{(i)}$ singular-values. The number of dimensions that is $\kappa_t^{(i)}$, are selected in such a way that the cumulative sum of these corresponding singular-values is greater than or equal to $\alpha/100$. Once, the size of the projected space is determined, a $\kappa_t^{(i)}$−rank approximation of SVD is derived to estimate the projection [50]. Based on the explanation given above, $\kappa_t^{(i)}$ can be mathematically written as

$$\kappa_t^{(i)} = min\{j : \sum_{m=1}^{j} \frac{\lambda_{t,m}}{\sum_{l=0}^{\eta_t^{(i)}} \lambda_{t,l}} \geq \frac{\alpha}{100}, j = 1, \cdots, \eta_t^{(i)}\}, \tag{11}$$

where $\eta_t^{(i)}$ denotes the total attributes in a group t at level (i). The low dimensional approximation corresponding to $\boldsymbol{C}_t^{(i)}$ is then computed [50] and denoted as $\hat{\boldsymbol{C}}_t^{(i)} = \hat{\boldsymbol{U}}_t^{(i)} \hat{\boldsymbol{\Sigma}}_t^{(i)} (\hat{\boldsymbol{V}}_t^{(i)})^T$. The term $\hat{\boldsymbol{U}}_t^{(i)}$ represents the singular-basis of rank $\kappa_t^{(i)}$ and $\hat{\boldsymbol{\Sigma}}_t^{(i)}$ represents a diagonal matrix with $\kappa_t^{(i)}$ singular-values.

For the current example, let $\alpha = 95\%$ and assume that the data-vectors are projected onto a one dimensional space. The singular-basis for the low dimensional approximation provides the transformation parameters to enable this projection. Finally, the process of evaluating the DC matrix, calculating the low dimensional approximation and projection of the data is completed for each group at step i.

Formally, let $\boldsymbol{P}_t^{(i)} = \hat{\boldsymbol{U}}_t^{(i)}$ denote the transformation for each group and derive a cumulative transformation as

$$\begin{aligned} \hat{\boldsymbol{X}}^{(i+1)} &= \{\boldsymbol{X}_t^{(i)} \boldsymbol{P}_t^{(i)}, t = 1, 2, \cdots T^{(i)}\} \\ &= [\boldsymbol{X}_1^{(i)} \boldsymbol{P}_1^{(i)}, \boldsymbol{X}_2^{(i)} \boldsymbol{P}_2^{(i)}, \cdots, \boldsymbol{X}_{T^{(i+1)}}^{(i)} \boldsymbol{P}_{T^{(i+1)}}^{(i)}] \\ &= \boldsymbol{X}^{(i)} \boldsymbol{P}^{(i)}, \end{aligned} \tag{12}$$

where $\boldsymbol{X}^{(i)} = [\boldsymbol{X}_1^{(i)} \quad \boldsymbol{X}_2^{(i)} \quad \cdots \quad \boldsymbol{X}_{T^{(i+1)}}^{(i)}]$ with $\boldsymbol{P}^{(i)} = diag(\boldsymbol{P}_1^{(i)}, \boldsymbol{P}_2^{(i)}, \cdots, \cdots \boldsymbol{P}_{T^{(i+1)}}^{(i)})$. For this illustrative example, four data-vectors are derived from the original data after the four groups at the first step are evaluated. The next step is to describe the stopping criterion.

The stopping criterion must evaluate the degree of information loss while quantifying the information provided by any dimension in the data. Since variance is considered as the measure of information, the quantification is provided by the singular-values. Specifically, an average of all the singular-values in a group quantifies the information retained in a group. Consequentially, one can compare this average value to $\alpha/100$ and halt the dimension reduction methodology, if the average is equal to or smaller than $\alpha/100$. The scenario implies that the cumulative average of all the dimensions in each group is exactly equal to $\alpha/100$ and any reduction in dimensions is going to make the average less than $\alpha/100$. The end result provides the information that the process of dimension reduction is optimal and the dimensions cannot be reduced further without additional information loss.

For our example, lets assume that the stopping criterion is not satisfied and re-initiate the process of grouping. Furthermore, it is assumed that the transformation stage ended with four dimensions in the data for our example.

3.3 Stage-1: Groupings at the Later Steps that is $i > 1$

Since, four dimensions are obtained at the end of the previous transformation step, these dimensions must be grouped into two groups and the strength of their respective singular-values can now be used to group these dimensions. However, these groupings must be performed in such a way that dimensions with larger singular-values are not grouped together. This way it can be ensured that the dimensions that imply large variances are not present in the same group and are not prematurely discarded.

With our example, let's say that the first two attribute contribute more to variance relative to the others. Therefore, one and two must be placed in separate groups which would mitigate the chance of premature loss of these attributes. With this line of thought, the first and the third attribute form the first group and the rest two attributes form the second group. In a general case, this can be accomplished by ordering the dimensions according to singular-values then grouping the first and the last dimensions together.

One can observe that the grouping mechanism is nothing but a shuffling mechanism and can be represented in matrix form. Let this matrix be defined as $\boldsymbol{Q}^{(i)}$ then the transformation between two successive steps can be rewritten as

$$\boldsymbol{X}^{(i+1)} = \boldsymbol{Q}^{(i)}\boldsymbol{X}^{(i)}\boldsymbol{P}^{(i)}.$$

Once the shuffling process is performed, one step of the dimension reduction process is finished, that is a grouping and a transformation stage. These stages are continued for I steps and a generalized expression for this transformation can be written as

$$\boldsymbol{X}^{(I)} = \boldsymbol{Q}^{(I)} \cdots \boldsymbol{Q}^{(1)}\boldsymbol{X}^{(1)}\boldsymbol{P}^{(1)} \cdots \boldsymbol{P}^{(I)} = \boldsymbol{Q}\boldsymbol{X}^{(1)}\boldsymbol{P}, \tag{13}$$

where $\boldsymbol{Q} = \boldsymbol{Q}^{(I)} \cdots \boldsymbol{Q}^{(1)}$ and $\boldsymbol{P} = \boldsymbol{P}^{(1)} \cdots \boldsymbol{P}^{(I)}$. Denoting this transformation as F, one can rewrite Eq. (13) as

$$\boldsymbol{X}^{(I)} = F(\boldsymbol{X}^{(1)}; \boldsymbol{Q}, \boldsymbol{P}). \tag{14}$$

where $F(.; \boldsymbol{Q}, \boldsymbol{P})$ is the dimension reducing transformation as illustrated in Fig. 3.

Remark 3 Through the use of NDR, efficient dimension reduction in large dimensional cases can be ensured. This is due to the fact that the problem of singular random matrices can be avoided through the use of the group-wise reduction strategy and by choosing the group sizes.

With our example, one may achieved a total of two dimensions as a result of two transformation steps with $I = 2$ which implies that there are only two useful dimensions in our example. By removing all the other dimensions from the data, the problem of unwanted dimensions is handled but the issue of heterogeneity remains which is described next.

4 Classification

For illustration in this section, the batch of data that has been reduced into two dimensions is utilized, that is $\boldsymbol{\mathcal{X}}_1$. Since, the complete picture of the underlying problem is not represented by the samples at hand, it is obvious that the learning model can encounter characteristics not described by the training data. Due to these unseen characteristics, errors may appear that can be quantified as generalization error J_{gen}.

However, these characteristics are unknown and the resulting error cannot be predicted beforehand. Therefore, an approximation of this error must be derived and unseen samples must be generated for this purpose. To achieve this approximation, let every data-point be the center of a neighborhood such that it is reasonable to assume that the data-points in the neighborhood belong to the same category as the center point. Next, generate samples from the neighborhood such that these additional samples can be seen as unseen samples providing an approximation of generalization error $\boldsymbol{e}_{gen}$ and its associated cost J_{gen}.

Formally, let $\Delta \boldsymbol{x} \sim p'(\boldsymbol{\mu}, \boldsymbol{\mathcal{S}})$ represent the perturbations introduced into every data-point $\boldsymbol{x}$ to generate the neighborhood. Let $p'(.)$ denote a probability distribution that can be fully described by the first two moments: mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\mathcal{S}}$. Perturb every data-point $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_1$ to get $\boldsymbol{x} + \Delta \boldsymbol{x}$ with $\|\Delta \boldsymbol{x}\| \leq Q$. A collection of all of these perturbed data-points denoted by $\boldsymbol{x} + \Delta \boldsymbol{x}$ represents the neighborhood for $\boldsymbol{x}$. It follows that $\boldsymbol{\mathcal{X}}_B = \{\boldsymbol{x}_B | \boldsymbol{x}_B = \boldsymbol{x} + \Delta \boldsymbol{x}, \forall \|\Delta \boldsymbol{x}\| \leq Q, \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}_1\}$. An approximation of generalization error can be obtained from the neighborhood that can be written as $E[\boldsymbol{e}_{gen}] = E_{\forall \boldsymbol{x}_B \in \boldsymbol{\mathcal{X}}_{1B}, \boldsymbol{y} \in \boldsymbol{\mathcal{Y}}_1}[\boldsymbol{y} - \hat{\boldsymbol{y}}]$.

In other words, for our example, an additional batch of data is generated in the two dimensions by perturbing the original data-batch $\boldsymbol{\mathcal{X}}_1$. Let the collection of these two data-batches be denoted as $\boldsymbol{\mathcal{X}}_{1B}$. An histogram of $\boldsymbol{\mathcal{X}}_{1B}$ for our example case is illustrated in Fig. 2a. It is observed that $\boldsymbol{\mathcal{X}}_{1B}$ represents Ψ better than $\boldsymbol{\mathcal{X}}_1$. This indicates that the impact of heterogeneity is compensated through the use of the additional samples.

Once this is done, the cost of generalization is incorporated into the learning problem to improve the resilience of the classifier. To this end, define $E[\boldsymbol{e}_{gen}]^T E[\boldsymbol{e}_{gen}]$ as $\hat{J}_{gen}$ and substitute into Eq. (6) to reveal

$$
\begin{aligned}
J(\hat{\boldsymbol{\theta}}) &= \frac{1}{2}[J_{emp} + \hat{J}_{gen}] + \quad J_{apx}, \\
J(\hat{\boldsymbol{\theta}}) &= \frac{1}{2}\left[E[\boldsymbol{e}_l] \quad E[\boldsymbol{e}_{gen}]\right]^T \left[E[\boldsymbol{e}_l] \quad E[\boldsymbol{e}_{gen}]\right] + \quad J_{apx},
\end{aligned}
\tag{15}
$$

where the learning error is denoted as $\boldsymbol{e}_l$ and the approximated generalization error is denoted as $\boldsymbol{e}_{gen}$. For brevity of notation, the expectation operators are not explicitly stated from hereon. By minimizing the new cost function, it is possible to minimize the empirical cost along with the approximated generalization cost. Thus, the learning problem is given as

$$\boldsymbol{\theta}^* = \underset{\hat{\theta} \in \boldsymbol{\Omega}}{\operatorname{argmin}} J(\hat{\boldsymbol{\theta}}), \tag{16}$$

with $\boldsymbol{\Omega}$ representing the parameter space.

Remark 4 Depending on the amount of perturbations that are introduced into the data, the neighborhood represents the entire input space. However if $\Delta \boldsymbol{x}$ is large, outliers are included in the neighborhood. On the other hand, small perturbations lead to large generalization errors.

Next, to optimize the cost function, an update law is defined as

$$\hat{\boldsymbol{W}}_{k+1}^{(i)} = \hat{\boldsymbol{W}}_{k}^{(i)} - \alpha \boldsymbol{u}_{k}^{(i)}, \tag{17}$$

with $\alpha > 0$ being the learning rate. Let weight decay be introduced as a constraint into the problem and write the Lagrangian as

$$H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \theta) = \frac{1}{2}\Big[J(\boldsymbol{\theta}) + \lambda \sum_{i=1}^{d} \|\boldsymbol{W}_k^{(i)}\|^2\Big], \tag{18}$$

with $\lambda > 0$ being the decay coefficient. The learning problem in Eq. (16) can be rewritten as

$$\boldsymbol{\theta}^* = \arg \min_{\hat{\theta}} H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \theta). \tag{19}$$

When learning is performed using gradient descent, the weight update for each layer at iteration k is given as

$$\boldsymbol{u}_k^{(i)} = \nabla_{\hat{\boldsymbol{W}}_k^{(i)}} H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \theta),$$

where $\nabla()$ represents the derivative. Finally, the updates for the weights are achieved as

$$\boldsymbol{u}_k^{(i)} = \boldsymbol{\delta}_k^{(i)} + \lambda \hat{\boldsymbol{W}}_k^{(i)}$$

and the term $\boldsymbol{\delta}_k^{(i)}$ is given as

$$\boldsymbol{\delta}_k^{(i)} = \nabla_{\hat{\boldsymbol{W}}_t^{(i)}} J = \boldsymbol{G}^{(i)}(\boldsymbol{x}) + \boldsymbol{G}^{(i)}(\boldsymbol{x} + \Delta \boldsymbol{x}), \tag{20}$$

with $G^{(i)}(\boldsymbol{x}) = \nabla_{\hat{\boldsymbol{W}}_t^{(i)}} J_{emp}$ being the derivative of the empirical cost and $G^{(i)}(\boldsymbol{x} + \Delta x) = \nabla_{\hat{\boldsymbol{W}}_t^{(i)}} \hat{J}_{gen}$ being the derivative of the generalization cost. Note that both the original data and the additional samples contribute to the learning signal. Simplifying through chain rule reveals

$$\boldsymbol{G}^{(i)}(\boldsymbol{x}) = -f^{(i-1)}(\boldsymbol{x})\boldsymbol{e}_l^T \Bigg[\prod_{j=d}^{i+1} diag(\nabla f^{(j)}(\boldsymbol{x}))\hat{\boldsymbol{W}}^{(j)}\Bigg] diag(\nabla f^{(i)}(\boldsymbol{x})), \tag{21}$$

Let $\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x}) = \prod_{j=d}^{i+1} diag(\nabla f^{(j)}(\boldsymbol{x}))\hat{\boldsymbol{W}}^{(j)}$ and simplify Eq. (20) to get

$$\boldsymbol{\delta}_k^{(i)} = -[\underbrace{f^{(i-1)}(\boldsymbol{x})}_{\eta^{(i-1)}\times 1} \quad \underbrace{f^{(i-1)}(\boldsymbol{x}+\Delta\boldsymbol{x})}_{\eta^{(i-1)}\times 1}]\boldsymbol{\epsilon}^T\boldsymbol{\mathcal{T}}^{(i)} \begin{bmatrix} \underbrace{(diag(\nabla f^{(i)}(\boldsymbol{x}))}_{\eta^{(i)}\times\eta^{(i)}} \\ \underbrace{diag(\nabla f^{(i)}(\boldsymbol{x}+\Delta\boldsymbol{x}))}_{\eta^{(i)}\times\eta^{(i)}} \end{bmatrix}, \tag{22}$$

where the overall error can be given as

$$\boldsymbol{\epsilon}^T = diag(\frac{dJ(\boldsymbol{\theta})}{d\hat{\boldsymbol{y}}})^T = diag(\underbrace{\boldsymbol{e}_l}_{\mathcal{F}\times 1}, \underbrace{\boldsymbol{e}_{gen}}_{\mathcal{F}\times 1}). \tag{23}$$

It follows that

$$\boldsymbol{\mathcal{T}}^{(i)} = diag(\underbrace{\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x})}_{\mathcal{F}\times\zeta^{(i)}}, \underbrace{\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x}+\Delta\boldsymbol{x})}_{\mathcal{F}\times\zeta^{(i)}}), \tag{24}$$

with $\zeta^{(i)}$ representing the number of neurons in the hidden layer (i).

As observed above, $\boldsymbol{\epsilon}$ is normalized by a transformation $\boldsymbol{\mathcal{T}}^{(i)}$ to impact learning with the traditional gradient descent. In other words, the linear approximations of the cost function that is $\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x})$ and $\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x}+\Delta\boldsymbol{x}))$ determine the learning directions and the magnitude of learning for the DNN. Furthermore, it can be observed that the directions of learning are influenced by the singular vectors of $\boldsymbol{\mathcal{T}}^{(i)}$ and singular values of $\boldsymbol{\mathcal{T}}^{(i)}$ describe the learning magnitude. Since, each diagonal element in $\boldsymbol{\mathcal{T}}^{(i)}$ is the products of the layer-wise activation function derivatives, it can be seen that $\|\boldsymbol{\mathcal{T}}^{(i)}\| \geq \sigma_{min}(\boldsymbol{\mathcal{T}}^{(i)})$, where $\sigma_{min}(\boldsymbol{\mathcal{T}}^{(1)})$ is the smallest singular value. If $\|\boldsymbol{\mathcal{T}}^{(1)}\| \Rightarrow 0$ then it is also true that $\sigma_{min}(\boldsymbol{\mathcal{T}}^{(i)}) \Rightarrow 0$, which one may get by applying squeeze theorem [36]. In simpler terms, with an increase in the number of layers in the DNN, the singular values of $\boldsymbol{\mathcal{T}}^{(i)}$ would diminish, which is also known as the vanishing gradients issue [36]. As a solution to the issue, one can utilize the error directly for learning the weights at every layer and this method is referred to as error-driven learning (EDL).

To enable such a learning state, a performance measure $H^{(i)}(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{W}}^{(i)})$ is described for each layer in the DNN and $H^{(i)}(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{W}}^{(i)})$ is optimized. Specifically, the layer-wise cost is defined in such a way that minimizing $H^{(i)}(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{W}}^{(i)})$ minimizes the overall cost. It follows that

$$H^{(i)}(\hat{\boldsymbol{W}}^{(i)}) = \frac{1}{2}\left[tr((\hat{\boldsymbol{W}}_k^{(i)})^T\boldsymbol{\delta}_k^{(i)}) + \lambda\|\boldsymbol{W}_k^{(i)}\|\right], \tag{25}$$

where $tr(.)$ is the trace operator. Observe that the cost function is defined in a such a way that $\boldsymbol{\delta}_k^{(i)}$ can be understood as a feedback of the cost function on layer i. It follows that by minimizing the cost-function, one would minimize the feedback which is basically the error. It can therefore be understood that each layer is independently contributing towards minimizing the error.

To define $\boldsymbol{\delta}_k^{(i)}$, at each layer with the new setup, $\boldsymbol{\mathcal{T}}^{(i)}$ is replaced by $\boldsymbol{B}^{(i)} = diag(\underbrace{\boldsymbol{B}^{(i)}(\boldsymbol{x})}_{\mathcal{F}\times\eta^{(i)}}, \underbrace{\boldsymbol{B}^{(i)}(\boldsymbol{x}+\Delta\boldsymbol{x})}_{\mathcal{F}\times\eta^{(i)}})$, where $\boldsymbol{B}^{(i)}$ is defined to satisfy positive definiteness of $\boldsymbol{B}^{(i)}(\boldsymbol{B}^{(i)})^T$. Then $\boldsymbol{\delta}_k^{(i)}$ is given as

$$\boldsymbol{\delta}_k^{(i)} = [\underbrace{f^{(i-1)}(\boldsymbol{x})}_{\eta^{(i-1)}\times 1} \quad \underbrace{f^{(i-1)}(\boldsymbol{x}+\Delta\boldsymbol{x})}_{\eta^{(i-1)}\times 1}]\boldsymbol{\epsilon}^T\boldsymbol{B}^{(i)}. \tag{26}$$

The overall cost can therefore be written as

$$H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \theta) = \sum_{i=1}^{d} H^{(i)}(\hat{\boldsymbol{W}}^{(i)}) \tag{27}$$

The updates for the weights are then given as

$$\boldsymbol{u}_k^{(i)} = \nabla_{\hat{\boldsymbol{W}}_k^{(i)}} H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \theta). \tag{28}$$

The success of the above described learning regime depends on the careful choice of $\boldsymbol{B}^{(i)}$ which is described next.

Choice of $\boldsymbol{B}^{(i)}$: One way to choose $\boldsymbol{B}^{(i)}$ is to sample at random from a pre-selected distribution. The main advantage of this is that it will enforce learning progression in random directions thus improving the exploration of the methodology [35]. On the other hand, this will provide the learning problem with no real navigation capabilities. Another way to choose is to ensure that the DNN learns in directions of steepest descent for $J(\theta)$. The construction of $\boldsymbol{B}^{(i)}$ decides the learning directions. Therefore, one can construct $\boldsymbol{B}^{(i)}$ in such a way that the learning is guided towards the minimum of the cost function. To do this, one may use the learning directions that are dictated by the singular vectors of $\boldsymbol{\mathcal{T}}^{(i)}$. With the size of the output vector in the DNN being $\mathcal{F}$ and the total number of hidden layers being denoted as $\eta^{(i)}$, one can observe that $\boldsymbol{\mathcal{T}}^{(i)} \in \mathbb{R}^{\mathcal{F}+\eta^{(i)}\times\mathcal{F}+\eta^{(i)}}$. Finally, one may write $\boldsymbol{B}^{(i)} = \sum_{j=1}^{\mathcal{F}+\eta^{(i)}} \kappa_j^{(i)}\boldsymbol{\nu}_j^{(i)}(\boldsymbol{\nu}_j^{(i)})^T$, where singular vectors are denoted by $\boldsymbol{\nu}_j$ and the singular values are denoted by $\kappa_j > 0$. The weight update rules for both gradient descent and EDL are summarized in Table 2.

With our example, the DNN parameters and the parameters of NDR have been updated for the first batch of data. From the overall approach described in Fig. 4, it can

Table 2 Summary of the batch-wise update laws with $\hat{\boldsymbol{W}}_{k+1}^{(i)} = \hat{\boldsymbol{W}}_k^{(i)} - \alpha\boldsymbol{u}_k^{(i)}$

	$(\boldsymbol{W}^{(i)})$
Gradient descent	$\boldsymbol{u}_k^{(i)} = E[\boldsymbol{\delta}_k^{(i)}] + \lambda\boldsymbol{W}_k^{(1)}$, where $\boldsymbol{\delta}_k^{(i)}$ is defined in Eq. (20)
EDL	$\boldsymbol{u}_k^{(i)} = E[\boldsymbol{\delta}_k^{(i)}] + \lambda_1 E[\nabla_{\boldsymbol{W}^{(i)}}\hat{H}^{(i)}] + \lambda_2\boldsymbol{W}_k^{(i)}$, where $\boldsymbol{\delta}_k^{(i)}$ is defined in Eq. (26)

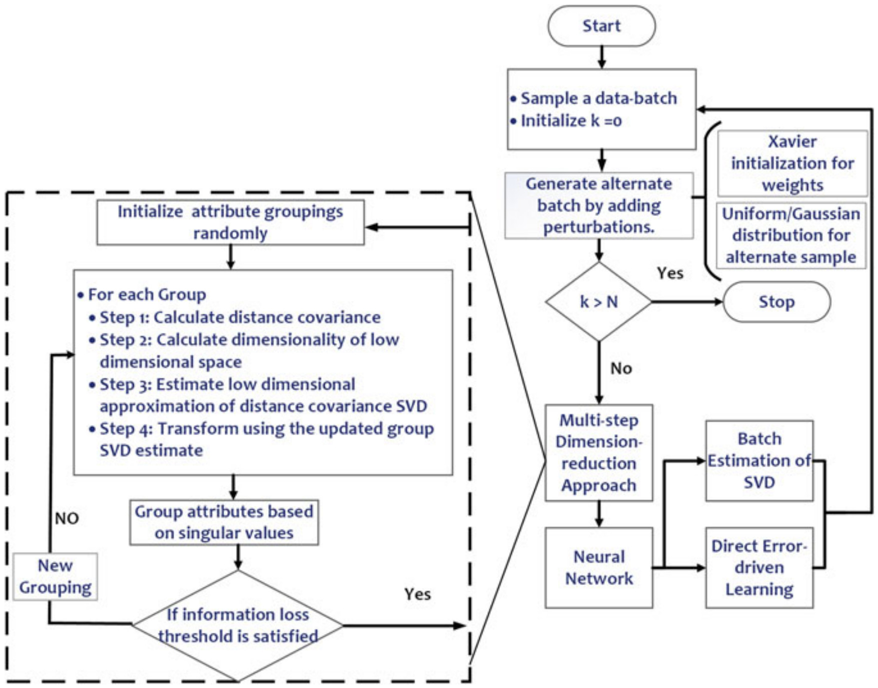

Start
• Sample a data-batch
• Initialize k =0
Generate alternate batch by adding perturbations.
Xavier initialization for weights
Uniform/Gaussian distribution for alternate sample
k > N
Yes
Stop
No
Multi-step Dimension-reduction Approach
Neural Network
Batch Estimation of SVD
Direct Error-driven Learning
Initialize attribute groupings randomly
• For each Group
• Step 1: Calculate distance covariance
• Step 2: Calculate dimensionality of low dimensional space
• Step 3: Estimate low dimensional approximation of distance covariance SVD
• Step 4: Transform using the updated group SVD estimate
Group attributes based on singular values
NO
New Grouping
If information loss threshold is satisfied
Yes

Fig. 4 Proposed two-step classification framework

be observed that the parameters of NDR and update for the DNN weights is performed for each batch of the data. Thus, there is a need to aggregate the information from different batches of the data otherwise the issue of heterogeneity will arise. Therefore, a process to aggregate the information across batches for both the DNN and NDR is described next.

5 Two-Step Classification Framework

A flow chart of the two-step classification framework is provided in Fig. 4. As illustrated in the figure, each batch of data is first transformed using NDR. Next, the transformed data is used to calculate cost function and the corresponding error signal is directly used for learning each layer in the DNN. The process is repeated for each batch in the data for several iterations. However, for the dimension reducing transformation, one would have to recalculate the parameters for each batch. To avoid this issue, batch-wise updates of the dimension reducing are introduced.

Batch-wise updates for the DNN weights are trivial because these have been well studied in the literature [29]. However, to aggregate the information across multiple batches for the dimension reduction approach, novel procedures need to be derived and these are described in [37]. The batch updates described in [37] follow the idea of updating an old estimate till now using the information from a combination between the new estimate and the old one. Following the notation from [37], one can write the batch wise update as

$$\boldsymbol{C}^{(q)} = \boldsymbol{C}^{(q-1)} + 2\hat{\boldsymbol{C}}^{(q-1)\times(q)} + \hat{\boldsymbol{C}}^{(q),(q)}, \tag{29}$$

with an aggregated estimate of distance covariance denoted as $\boldsymbol{C}^{(q)}$ where all the batches till batch b_{q-1} is included. Let $\hat{\boldsymbol{C}}^{(q-1)\times(q)}$ be an estimate from the data (that is batch b_q), along with the previously evaluated estimate. Furthermore, $\hat{\boldsymbol{C}}^{(q),(q)}$ provides the estimate from the current batch b_q. Similarly, the SVD updates can be also written as

$$\boldsymbol{U}^{(q)} = [\boldsymbol{U}^{(q-1)} \quad I]\hat{\boldsymbol{U}}^{(q)}, \boldsymbol{V}^{(q)} = [\boldsymbol{V}^{(q-1)} \quad I]\hat{\boldsymbol{V}}^{(q)}, \quad \boldsymbol{\Sigma}^{(q)} = \hat{\boldsymbol{\Sigma}}^{(q)}, \tag{30}$$

where $\hat{\boldsymbol{U}}^{(q)}\hat{\boldsymbol{\Sigma}}^{(q)}(\hat{\boldsymbol{V}}^{(q)})^T = SVD(\boldsymbol{K})$ and $\boldsymbol{K} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}}^{(q)} & \hat{\boldsymbol{\Sigma}}^{(q)}(\hat{\boldsymbol{V}}^{(q-1)})^T \boldsymbol{J}^{(q)} \\ 0 & \boldsymbol{R}_{\boldsymbol{J}}^{(q)} \end{bmatrix}$. Next, $\hat{\boldsymbol{U}}^{(q)}$, $\hat{\boldsymbol{\Sigma}}^{(q)}$ and $\hat{\boldsymbol{V}}^{(q)}$ are decomposition from $\boldsymbol{K}^{(q)}$ with $\boldsymbol{J}^{(q)} = 2\hat{\boldsymbol{C}}^{(q-1),(q)} + \hat{\boldsymbol{C}}^{(q),(q)}$ and $\boldsymbol{R}_{\boldsymbol{J}}^{(q)} = (\boldsymbol{J}^{(q)} - \hat{\boldsymbol{V}}^{(q-1)}(\hat{\boldsymbol{V}}^{(q-1)})^T \boldsymbol{J}^{(q)})$. It could be seen that the middle term in the batch-wise updates requires that a representative batch sample for all the data till now be kept. Theoretically, it can be shown that if an infinite number of data-points are available then the estimated DC and its SVD will converge to the true values (see Theorem 1 in [37]). Since, this is not possible in practical cases, a finite history generated using popular sampling techniques would suffice.

The efficiency of the two step approach is presented next.

Algorithm 1 Low-rank approximation for SVD, j represents the dimension in the matrix, d represents the singular value

```
1:  Inputs: matrix $\boldsymbol{K}$, $\kappa = \kappa_t^{(i)}; \forall t, i$.
2:  Outputs: $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$.
3:  Let $C_1 = C$.
4:  Let $\kappa_1 = 0$.
5:  for $j \in 1, 2, ..., \kappa$.
6:      for Iterate until convergence, $\forall k$.
7:          $\boldsymbol{u}_k \leftarrow \frac{(\boldsymbol{K}^T\boldsymbol{K})^k \boldsymbol{u}_0}{\|(\boldsymbol{K}^T\boldsymbol{K})^k \boldsymbol{u}_0\|}$
8:          $\boldsymbol{u}_k \leftarrow \frac{(\boldsymbol{K}^T\boldsymbol{K})^k \boldsymbol{v}_0}{\|(\boldsymbol{K}^T\boldsymbol{K})^k \boldsymbol{v}_0\|}$.
9:      end for
10:     $d_j \leftarrow \boldsymbol{u}_j^T \boldsymbol{C}_j \boldsymbol{v}_j$
11:     $\boldsymbol{C}_j = \boldsymbol{C}_{j-1} - d_j \boldsymbol{u}_j^T \boldsymbol{v}_j$.
12: end for
13: Final SVD, $\boldsymbol{U} = \{\boldsymbol{u}_j\}, \boldsymbol{V} = \{\boldsymbol{v}_j\}, \boldsymbol{\Sigma} = diag(\{d_j\}), \forall j = 1, 2, \cdots, \kappa$.
```

6 Results and Discussion

The two step classification framework is employed to analyze a total of ten data-sets in this section and the details of the data-set are described in Table 3. The split for the training and test data-set is done at random. Twenty percent of the data-set is used for test and the rest eighty percent is used for training. The results are summarized using mean and variance. The software package Tensor-flow with Python is used for all of the experiments in this paper. The advantages of the NDR in big-data situations is demonstrated first.

Algorithm 2 A Nonlinear dimension-reduction (NDR)

```
1:  Input: $\boldsymbol{\mathcal{X}}$, $\alpha$
2:  Output:Reduced dimension $\boldsymbol{\mathcal{X}}$
3:
4:  for Each $batch$ in $\boldsymbol{\mathcal{X}}$
5:      $\boldsymbol{X}^{(1)} = batch$
6:      Standardize $\boldsymbol{X}^{(1)}$.
7:      for step from $i = 1 \rightarrow I$
8:          if $\frac{1}{T^{(i)}} \sum_{t=0}^{T^{(i)}} \sum_{l=0}^{\kappa} \lambda_l \geq \alpha$
9:              Stop the dimension-reduction procedure
10:         end if
11:         if i>1
12:             Create groups
13:         else
14:             Generate groupings at random
15:         end if
16:         for Every group at step $i$
17:             Calculate Distance Correlation matrix
18:             if batch number is one
19:                 Use current Distance Correlation
20:                 Evaluate $\kappa$
21:             else
22:                 Use the aggregated Distance Correlation
23:                 Evaluate $\kappa$
24:             end if
25:             Evaluate low rank approximation using Alg. 1
26:             Update distance covariance using Eq (29)
27:             Update low rank SVD using Eq (30).
28:         end for
29:     end for
30:     Train the regression parameter using Alg. 2
31: end for
```

Table 3 The statistics of different data-sets used in the paper

Data-set	Dimensions	Data points	Classes
Rolling [43]	11	35000	4
Sensorless [31]	48	78000	11
MNIST [28]	784	72000	10
NotMNIST [5]	784	81000	10
CiFAR-10 [26]	3072	50000	10
Gisette [18]	5000	6000	2
Madelon [18]	500	1000	2
Arcene [18]	10000	100	2
Dexter [18]	20000	300	2
Image-Net [40]	784	12614060	10
Synthetic data-set	20000	2000000	10

6.1 Mitigating Noisy Dimension

The hyper parameters for various methodologies are described in Table 4 with the results for sensorless drive-diagnostics data-set tabulated in Table 5. The results are seen to be optimal with Type-1 error rate increasing with an increase in the number of dimensions extracted from the data. The best accuracy is observed when the dimensions are reduced to 35. Next, the results for NDR are compared with other approaches and the results are summarized in Table 9.

Better performance is observed for NDR relative to other approaches. As, sensorless drive data-set typically consists of nonlinear relationships, PCA will fail, which is observed in the results. In these tests, the data is reduced to 25 and improvement is

Table 4 Hyper-parameters for different methodologies in this paper

Method	Hyper-parameters
K Nearest Neighbors (KNN)	Three neighbors
Support Vector Machine (SVM)	C = 0.025
Kernel SVM	RBF kernel, gamma = 2, C = 1
Decision Trees (DT)	Max depth = 5
Random Forest (RF)	Max depth = 5 estimators = 10, max attributes = 1
Shallow Neural Network (SNN)	Lr = 0.01, 1 hidden layer, 500 neurons
AdaBoost	Number of estimators = 50, learning rate = 1.0
Naive-Bayes	No priors
Quadratic Discriminant Analysis (QDA)	No priors
Logistic Regression (LR)	Random initialization, learning rate = 0.001, 1000 iterations

Table 5 Accuracies at various dimensions along with false positive rates for rolling element bearing data-set

$\alpha/100$	Dimensions	False positive rates	Accuracy
0.50	5	0.008	0.93
0.60	6	0.009	0.94
0.70	7	0.011	0. 96
0.95	10	0.011	0.98
0.99	11	0.011	0.95

Table 6 Average accuracies for the toy data-set over 100 runs

Methodology	With neighborhood	Without neighborhood
Half-Moons	93	84
Concentric circles	95	91

observed in large dimensional cases. Next, the results on the problem of heterogeneity are illustrated.

6.2 *Mitigating Heterogeneity*

To generate the neighborhood, Gaussian-distributed and uniformly-distributed perturbations are used where the mean vector is chosen as a zero and $\rho \boldsymbol{I}$ denotes the variance-covariance matrix. Let $\boldsymbol{I}$ represent an appropriate identity matrix whereas for the uniformly distributed perturbation choose $[-\rho, \rho]$ as parameters. The quantity ρ controls the magnitude of the variance and denotes the size of the neighborhood and one can enforce neighborhood size using $\Delta \boldsymbol{x} = \frac{\Delta \boldsymbol{x}}{\|\Delta \boldsymbol{x}\|}\rho$ with ρ chosen by the practitioner.

First, the results for EDL are presented with $\boldsymbol{B}^{(i)}$ being chosen through the singular vectors of $\boldsymbol{\mathcal{T}}^{(i)}$. The hyper-parameters for the proposed framework are summarized in Table 10, where hln is the number of hidden layer neurons; n_{neigh} denotes the number of neighborhood points.

The intensity of noise in the data is increased and the change in accuracy is demonstrated in Fig. 6b. The size of the neighborhood is kept as one and the average accuracies and generalization error over one hundred initial conditions of weights are plotted in Fig. 6b. The deterioration in performance with the proposed framework is smaller relative to the case without the perturbations. Therefore, an improvement in the resilience of DNN's is observed.

Using toy data-sets such as concentric circles and half moons data-set, one can observe the ability of the neighborhood principle to improve generalization and increase the resilience of neural networks. In Fig. 5a, one may observe that the data-

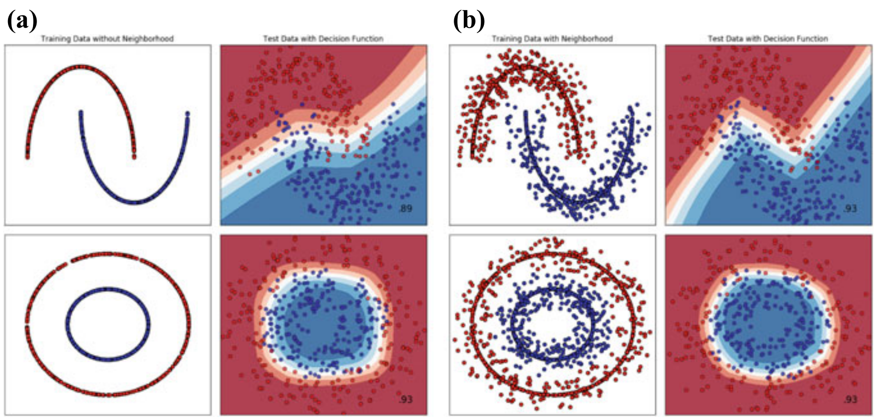
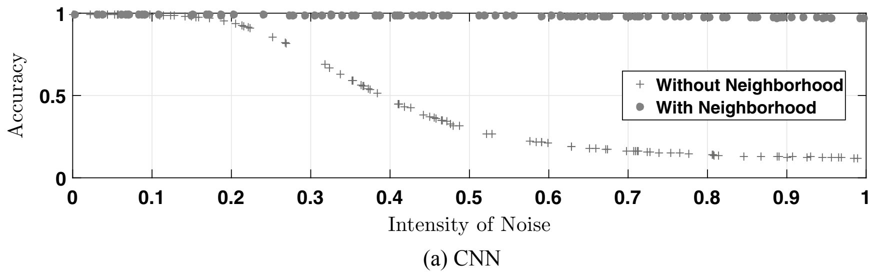
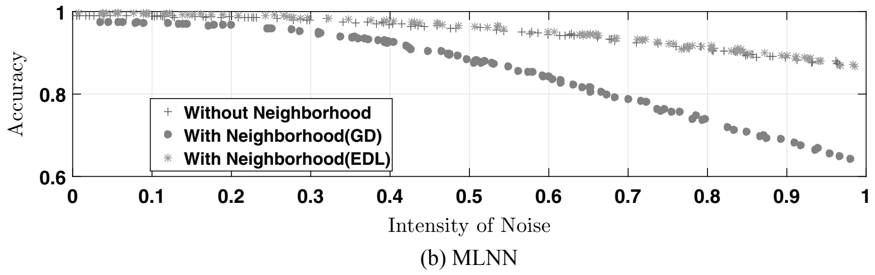


Fig. 5 **a** Decision Function for half moons data-set (top right) and concentric circles data-set (bottom right) without neighborhood. **b** Decision function for half moons data-set (top right) and concentric circles data-set (bottom right) with neighborhood





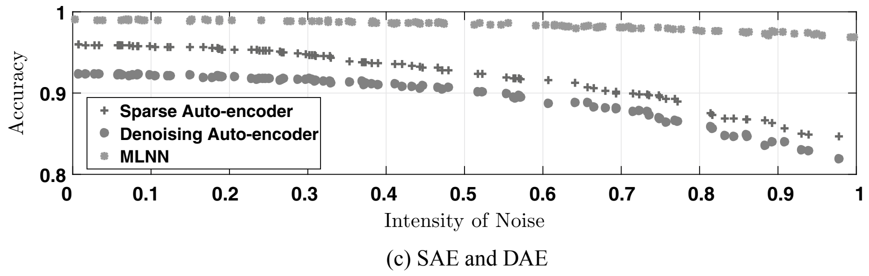


Fig. 6 Accuracy versus the increase in the intensity of noise in the data

Table 7 Model parameters

Architecture	Parameters
CNN	$n_{conv} = 2$, $n_{layers} = 2$, filter size $= 5$, pool shape $= (3,3)$, hln$= 128$, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$ $n_{neigh} = 2$, $\rho = 1$
MLNN	$d = 7$, hln $= 128$, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$ $n_{neigh} = 2$, $\rho = 1$
SAE	$n_{units} = 3$, hln$= 128$, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$ $n_{neigh} = 2$, $\rho = 1$
DAE	$n_{units} = 3$, hln$= 128$, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$ $n_{neigh} = 2$, $\rho = 1$

points closer to the decision boundary get misclassified in the absence of neighborhood. This may be observed from Table 6, the increase in accuracy with or without the neighborhood is shown for these data-sets.

It was also observed during our study that increasing the neighborhood size lets the perturbations overwhelm the signal. In other words, different categories in the data became indistinguishable and the performance of the proposed framework deteriorates significantly. As described by the plots in Fig. 5a, it is seen that when a data-point crosses the decision boundary due to the perturbations, it violates the primary assumption that the data-points in the neighborhood should belong to the same category as the original data and therefore errors are observed.

Next, convolutional neural networks (CNN) [27] and multilayer neural networks (MLNN) are tested using the proposed framework where the hyper-parameters are illustrated in Table 7. It is consistently observed across the board that there is an improvement in performance when perturbations are introduced to construct the neighborhood.

Next, sparse auto-encoders [48] and denoising auto-encoders [52] are used for analysis and one may refer to Table 7 for hyper-parameters. From Fig. 6c, one may observe an improvement in accuracy for the proposed framework over SAE's.

In summary, with the proposed approach, the resilience of DNN in the presence of heterogeneity and noise is improved. THe results are consistent across different architectures. The final results of the overall approach are described as follows.

6.3 *Classification*

The learning rates are chosen as 0.001 and the results are tabulated in Table 8. The resulting accuracy with dimension reduction is compared to the case without the dimension reduction process. With an increase in the number of dimensions in the

Table 8 Accuracies for different data-sets with and without the dimension-reduction

Data-set	With dimension-reduction	Without dimension-reduction
Arcene	**0.86**	0.73
Dexter	**0.99**	0.69
Gisette	**0.99**	0.71
Madelon	**0.85**	0.79
MNIST	**0.94**	0.89

Table 9 Accuracies for Sensorless Drive Diagnostics data set with comparison to different dimension-reduction approaches

Dim-Red ↓	**KNN**	SVM	LDA	DT	RF	SNN	AdaBoost	Naive-Bayes	QDA
PCA	0.52	0.52	0.523	0.4538	0.4529	0.5254	0.1844	0.45670	0.5377
ISOMAP	0.6752	0.4691	0.4302	0.3255	0.3975	0.5847	0.1844	0.4256	0.4925
LLE	0.2574	0.0876	0.1540	0.1886	0.1889	0.0876	0.1858	0.1911	0.1931
KPCA	0.54	0.53	0.523	0.4538	0.4488	0.5827	0.280	0.45670	0.5377
Lasso	0.2574	0.0876	0.1540	0.1886	0.1889	0.0876	0.1858	0.1911	0.1931
Elastic Net	0.54	0.53	0.523	0.4538	0.4488	0.5827	0.280	0.45670	0.5377
Proposed	**0.89**	**0.91**	**0.84**	**0.70**	**0.65**	**0.93**	**0.25**	**0.73**	**0.86**

data, the difference in accuracy increases and the case with dimension reduction provides better results that the alternative case. Furthermore, the proposed methodology is optimal even when the total number of data-points are less than the total number of dimensions. This is observed as the results on Arcene, Gisette and Dexter data-set show considerable improvement. The proposed methodology can be compared for accuracies with the most traditional classification methods and the results as described in Table 9. It is again seen that the proposed methodology performs optimally for data-sets such as Arcene, Gisette and Dexter data-set (Table 9).

Next, the performance of the proposed method is tested on classification. A total of four learning paradigm that include DFA (Direct Feedback Alignment) [35], SGD (Stochastic Gradient Descent) [19], FA (Feedback Alignment) [32] and EDL (Error-driven Learning), are tested. The hyper-parameters for different models are given in Table 10 that includes all the data-sets. Proposed framework appears to improve performance across the data-sets and reasonable accuracies are observed for the large dimensional data-sets.

Generalization capability of the proposed framework is described for different data-sets in Table 10. Lowest generalization error is observed for EDL and one can observe improved generalization errors in the presence of heterogeneity and noise.

Table 10 Accuracies with generalization error in parenthesis with different learning approaches. Hln refers to the number of hidden layer neurons at each layer and b denotes the batch size. Moreover, n_{neigh} represents the number of data-points in the neighborhood corresponding to each data-point in the data-set

Data-sets	p	n	$\mathcal{F}$	Model Parameters	DFA	FA	SGD	EDL
Rolling [43]	11	35000	4	$\alpha = 0.01$, $b = 100$, $d = 7$, $hln = 100$, $\lambda = 0.001$,$n_{neigh} = 2$, $\rho = 1$	99(0.0001)	99(0.0001)	99(0.6)	99(0.0)
Sensorless [31]	48	78000	11	$\alpha = 0.01$, $b = 100$, $d = 8$, $hln = 100$, $\lambda = 0.001$,$n_{neigh} = 2$, $\rho = 1$	94(0.008)	94(0.008)	95(0.008)	94(0.002)
MNIST [28]	784	72000	10	$\alpha = 0.01$, $b = 100$, $d = 8$, $hln = 100$, $\lambda = 0.001$,$n_{neigh} = 2$, $\rho = 1$	92(0.1)	94(0.2)	95(1)	97(0.6)
NotMnist [5]	784	81000	10	$\alpha = 0.01$, $b = 100$, $d = 9$, $hln = 100$ $\lambda = 0.001$,$n_{neigh} = 2$, $\rho = 1$	90(1)	82(1.8)	88(2)	92(0.8)
CIFAR10 [26]	3072	50000	10	$\alpha = 0.01$, $b = 100$, $d = 10$, $hln = 100$ $\lambda = 0.001$,$n_{neigh} = 2$, $\rho = 1$	81(7.21)	80(8.00)	82(12.11)	84(8.77)
Arcene [18]	10000	100	2	$\alpha = 0.01$, $b = 100$, $d = 9$, $hln = 100$, $\lambda = 0.001$ $\lambda = 0.001$,$n_{neigh} = 2$, $\rho = 1$	69(0)	61(0)	51(0)	78(0)
Dexter [18]	20000	300	2	$\alpha = 0.01$, $b = 100$, $d = 8$, $hln = 100$ $\lambda = 0.001$,$n_{neigh} = 2$, $\rho = 1$	71(0.7)	77(0.9)	81(1.1)	80(1)
Gisette [18]	5000	6000	2	$\alpha = 0.01$, $b = 100$, $d = 8$, $hln = 100$, $\lambda = 0.001$, $n_{neigh} = 2$, $\rho = 1$	92(0.25)	93(0.21)	91(0.3)	98(0.07)

7 Conclusions and Future Work

A systematic approach was presented to design a classifier in the presence of challenges such as heterogeneity, vanishing gradients and noise. Since synthetic distortions are introduced to approximate the generalization error, our framework is robust in the presence of data-noise because less variation in accuracy is observed. Careful selection of the magnitude of the perturbation is necessary because large perturbations deteriorate the performance of the proposed framework significantly.

The direct use of the overall error signal in the learning process appears to mitigate the vanishing gradient issue even when sigmoid and tanh activation functions were used. The proposed framework combined with EDL appears to outperform other approaches such as DFA and FA in all of the data-sets under analysis. Moreover, unwanted dimensions appear to impact accuracy less with the use of the dimension reduction approach.

NDR achieves very good accuracies in the presence of large dimensional data-sets. By controlling information loss, NDR can determine the cardinality of the low-dimensional space. As a result, the presence of unwanted dimensions can be effectively addressed within NDR.

One of the primary drawback of the two step approach is that the NDR does not get any feedback from the classification. This could be addressed through a comprehensive approach where the parameters of NDR are updated using the classification error. Investigating this issue could be part of future work.

References

1. Adragni, K.P., Al-Najjar, E., Martin, S., Popuri, S.K., Raim, A.M.: Group-wise sufficient dimension reduction with principal fitted components. Comput. Stat. **31**(3), 923–941 (2016)
2. Balasubramanian, M., Schwartz, E.L.: The isomap algorithm and topological stability. Science **295**(5552), 7 (2002)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
5. Bulatov, Y.: Notmnist dataset. Google (Books/OCR), Technical Report. http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html (2011)
6. Clarke, R., Ressom, H.W., Wang, A., Xuan, J., Liu, M.C., Gehan, E.A., Wang, Y.: The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. Nat. Rev. Cancer **8**(1), 37–49 (2008)
7. David, S., Ruey, S., et al.: Independent component analysis via distance covariance. J. Am. Stat. Assoc. (2017)
8. Donoho, D.L., Grimes, C.: Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. Proc. Natl. Acad. Sci. **100**(10), 5591–5596 (2003)
9. Fan, J., Han, F., Liu, H.: Challenges of big data analysis. Natl. Sci. Rev. **1**(2), 293–314 (2014)
10. Feng, H.: et al.: Gene classification using parameter-free semi-supervised manifold learning. IEEE/ACM Trans. Comput. Biol. Bioinform. **9**(3), 818–827 (2012)
11. Fodor, I.K.: A survey of dimension reduction techniques (2002)
12. Giraud, C.: Introduction to High-Dimensional Statistics, vol. 138. CRC Press (2014)

13. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)
14. Goldberger, J., Ben-Reuven, E.: Training deep neural-networks using a noise adaptation layer (2016)
15. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, vol. 1. MIT Press, Cambridge (2016)
16. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2014). *arXiv preprint* arXiv:1412.6572
17. Guo, Z., Li, L., Lu, W., Li, B.: Groupwise dimension reduction via envelope method. J. Am. Stat. Assoc. **110**(512), 1515–1527 (2015)
18. Guyon, I., Gunn, S., Ben-Hur, A., Dror, G.: Result analysis of the nips 2003 feature selection challenge. In: Advances in Neural Information Processing Systems, pp. 545–552 (2005)
19. Hardt, M.: 3.12 train faster, generalize better: stability of stochastic gradient descent. Math. Comput. Found. Learn. Theor. **64** (2015)
20. Ing, C.K., Lai, T.L., Shen, M., Tsang, K., Yu, S.H.: Multiple testing in regression models with applications to fault diagnosis in big data era. Technometrics (just-accepted) (2016)
21. Johnson, R.A., Wichern, D.W.: Applied Multivariate Statistical Analysis, vol. 4. Prentice Hall, Englewood Cliffs (1992)
22. Johnson, R.A., Wichern, D.W.: Applied Multivariate Statistical Analysis, vol. 5. Prentice Hall, Upper Saddle River (2002)
23. Jolliffe, I.: Principal Component Analysis. Wiley Online Library (2002)
24. Khan, F., Kari, D., Karatepe, I.A., Kozat, S.S.: Universal nonlinear regression on high dimensional data using adaptive hierarchical trees. IEEE Trans. Big Data **2**(2), 175–188 (2016)
25. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). *arXiv preprint* arXiv:1412.6980
26. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
28. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
29. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
30. Lee, D.H., Zhang, S., Fischer, A., Bengio, Y.: Difference target propagation. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 498–515. Springer, Cham (2015)
31. Lichman, M.: UCI machine learning repository. http://archive.ics.uci.edu/ml (2013)
32. Lillicrap, T.P., Cownden, D., Tweed, D.B., Akerman, C.J.: Random synaptic feedback weights support error backpropagation for deep learning. Nat. Commun. **713276** (2016)
33. Mishkin, D., Matas, J.: All you need is a good init (2015). *arXiv preprint* arXiv:1511.06422
34. Niyogi, P., Girosi, F.: On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. Neural Comput. **8**(4), 819–842 (1996)
35. Nøkland, A.: Direct feedback alignment provides learning in deep neural networks. In: Advances in Neural Information Processing Systems, pp. 1037–1045 (2016)
36. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318 (2013)
37. Krishnan, R., Samaranayake, V.A., Jagannathan, S.: A multi-step nonlinear dimension-reduction approach with applications to big data. IEEE Trans. Knowl. Data Eng. (2018)
38. Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., Rabinovich, A.: Training deep neural networks on noisy labels with bootstrapping (2014). *arXiv preprint* arXiv:1412.6596
39. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
40. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)

41. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
42. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Nello (2004)
43. Soylemezoglu, A., Jagannathan, S., Saygin, C.: Mahalanobis taguchi system (MTS) as a prognostics tool for rolling element bearing failures. J. Manuf. Sci. Eng. **132**(5), 051014 (2010)
44. Soylemezoglu, A., Jagannathan, S., Saygin, C.: Mahalanobis-taguchi system as a multi-sensor based decision making prognostics tool for centrifugal pump failures. IEEE Trans. Reliab. **60**(4), 864–878 (2011)
45. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
46. Sun, K., Huang, S.H., Wong, D.S.H., Jang, S.S.: Design and application of a variable selection method for multilayer perceptron neural network with LASSO. IEEE Trans. Neural Netw. Learn. Syst. **28**(6), 1386–1396, June 2017. ISSN 2162-237X. https://doi.org/10.1109/TNNLS.2016.2542866
47. Székely, G.J., Rizzo, M.L., Bakirov, N.K., et al.: Measuring and testing dependence by correlation of distances. Ann. Stat. **35**(6), 2769–2794 (2007)
48. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**(Dec), 3371–3408 (2010)
49. Ward, A.D., Hamarneh, G.: The groupwise medial axis transform for fuzzy skeletonization and pruning. IEEE Trans. Pattern Anal. Mach. Intell. **32**(6), 1084–1096 (2010)
50. Witten, D.M., Tibshirani, R., Hastie, T.: A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics **10**(3), 515–534 (2009)
51. Wu, X., Zhu, X., Wu, G.Q., Ding, W.: Data mining with big data. IEEE Trans. Knowl. Data Eng. **26**(1), 97–107 (2014)
52. Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. In: Advances in Neural Information Processing Systems, pp. 341–349 (2012)
53. Xu, N., Hong, J., Fisher, T.C.: Generalization error minimization: a new approach to model evaluation and selection with an application to penalized regression (2016). *arXiv preprint* arXiv:1610.05448
54. Yu, Z., Li, L., Liu, J., Han, G.: Hybrid adaptive classifier ensemble. IEEE Trans. Cybern. **45**(2), 177–190 (2015). ISSN 2168-2267. https://doi.org/10.1109/TCYB.2014.2322195
55. Zhang, L., Lin, J., Karim, R.: Sliding window-based fault detection from high-dimensional data streams. IEEE Trans. Syst. Man Cybern. Syst. (2016)
56. Zhou, J.K., Wu, J., Zhu, L.: Overlapped groupwise dimension reduction. Sci. China Math. **59**(12), 2543–2560 (2016)

Deep Learning for Soft Sensor Design

Salvatore Graziani and Maria Gabriella Xibilia

Abstract Soft Sensors are mathematical models used to predict the behavior of real systems. They are usefully applied to estimate hard-to-measure quantities in the process industry. Many Soft Sensors are designed by using data-driven approaches and exploiting historical databases. Machine learning is widely used for this aim. Here, the potentialities of deep learning in solving some challenges raising in industrial applications are introduced. More specifically, the paper focuses on three specific aspects: labelled data scarcity, computational complexity reduction, and unsupervised feature exploitation. The state of the art of Soft Sensors based on deep learning is described. Then, the focus is on Soft Sensors based on Deep Belief Networks, as a research field that the authors have been investigating since years. The improvements offered by Deep Belief Networks, over more conventional data-driven approaches, in designing Soft Sensors for real-world applications will be shown. Soft Sensors for specific cases study are described.

Keywords Soft sensors · System identification · Deep learning · Deep belief networks · Semi-supervised learning · Industrial applications

1 Soft Sensors in the Process Industry

The implementation of Industry 4.0 enforces the need for processes monitoring. Plant monitoring, control policies, and assets management impose the estimation of plant working conditions. Efficient measurement and data elaboration systems are, therefore, required. Ubiquitous sensing systems are required, with somewhat contrasting constraints on sensing capability, efficiency, redundancy, and costs. Physical

S. Graziani (✉)
DIEEI, University of Catania, Catania, Italy
e-mail: salvatore.graziani@dieei.unict.it

M. G. Xibilia
Department of Engineering, University of Messina, Messina, Italy
e-mail: mxibilia@unime.it

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_2

or economic reasons, e.g., can limit the nature, number, and location of measuring systems.

Models of real-world processes, devoted to the estimation of process variables by exploiting their dependence on other variables, are known as Soft Sensors (SSs). They are software tools that elaborate data on easy-to-measure process variables (SS inputs) and estimate hard-to-measure quantities (SS outputs).

SSs are becoming a penetrating solution in the industry, capable of overcoming physical limitations. They can face the problems mentioned above and are the object of a vivid interest by several communities, including industries, because of the following interesting properties:

- represent an alternative to hardware measuring devices;
- can be useful in measurement validation and fault detection;
- can be easily implemented, by using largely available and powerful hardware;
- allow for real-time estimation of variables, thus guaranteeing the possibility of implementing feedback control strategies.

The using of SSs in industrial applications and, in particular, in the process industry, has been widely addressed in the literature. The first contributions reviewing industrial case studies can be found in [1, 2]. More recently, many other cases study have been proposed in the literature. The discussion of all available literature is beyond the scope of this chapter. The focus of this chapter is, on the contrary, on specific issues arising in the SSs design and how they can be approached by deep learning.

Due to the complex and nonlinear nature of industrial processes, the model design is not a trivial issue and data-driven system identification procedures are widely proposed in the literature, as the most suitable approaches [1, 2]. Data extracted from a historical database, or purposefully collected during dedicated experimental measurement surveys, are used to this aim. Data-driven SSs are generally based on Principal Component Regression, Neural Networks (NNs), Fuzzy Logic, Kernel Learning Methods, and Bayesian approaches, to mention a few [2, 3].

Notwithstanding the interest in SSs, unsolved problems have so far hindered the full success of the data-driven designing approach. Recently, deep learning has emerged as a valuable approach for alleviating some of such problems. One of the main problems of NN based SSs is their inefficiency in representing very complex nonlinear phenomena, on the basis of a limited set of available experimental data. Though more complex representations can be obtained by using deeper structures, it has been shown that these can not be successfully trained by conventional BP-like algorithms, due to the vanishing and exploding gradient problems. The solution was given by the seminal works by Hinton and coworkers [4–6], who introduced a greedy layer-wise unsupervised pre-training and supervised fine-tuning, capable of efficiently training deep structures. As a consequence, the popularity of deep structures has continuously raised and now they represent the state of the art solutions in applications fields such as image processing, speech recognition, natural language processing, etc. [5]. A variety of deep structures have been proposed in the last years, including Stacked AutoEncoders (SAEs) [7], Deep Belief Networks (DBNs) [8], and Convolutional Neural Networks (CNNs) [9, 10].

The complexity of the nonlinear model representation is not the one problem in SS design that needs to be faced with. In a typical scenario of SS design for the process industry, physical process variables are sampled at a much faster rate than corresponding chemical quality variables. Process variables (flows, pressures, temperature, etc.) are, in fact, measured online (typically at one sample per minute or even at a faster rate), while quality variables are, generally, measured by laboratory equipment, typically, once or twice a day. This aspect represents a relevant problem when data-driven models need to be designed, based on available historical plant databases. If traditional machine-learning methods are used in the design of SSs for the process industry, only a very low fraction of available data (i.e., labelled data) can be exploited. The use of semi-supervised learning approaches, typical of deep learning, can give a meaningful solution to the described problem.

Deep structures also allow for approaching the issue of a large number of free parameters, with respect to available experimental data, often encountered in data-driven SSs, which poses the design at risk of overfitting. In fact, they allow for a compact representation of complex problems. Finally, the unsupervised nonlinear feature extraction process, which characterizes SAEs and DBNs training, can be usefully adopted to gain insight on the process structure, better driving the SS design process.

Many deep structures have been proposed for SS design, as it is described in the next section. The remaining part of the chapter focuses on the potentialities and challenges of DBNs in SSs design. Real-world industrial cases of study are described with the aim of enlighten how DBNs can approach the problems mentioned above. In details, the chapter is organized in such a way to focus on three different aspects of SS design with DBNs: model complexity reduction, exploitation of unlabelled data to improve the SS performance, and elaboration of dataset features for model structure selection.

2 State of the Art of SSs Based on Deep Learning

In the following, the state of the art of the use of deep networks in SS design is reported. The referred papers are organized based on the adopted deep network structure. The first network topology considered is the SAE. An SAE is a NN, consisting of a stack of autoencoders. Each autoencoder is a one hidden layer NN. The training of each autoencoder is performed by setting the output equal to the input. The autoencoder is required to accurately reconstruct its input, eventually removing noise from the data and/or mapping data into a lower dimensional manifold. An SAE is trained by performing a layer-wise unsupervised pre-training, followed by a supervised fine-tuning. During the pre-training step, each autoencoder maps its input data into the hidden feature layer. The gradient descent minimization of the reconstruction error is used to determine the weights. After an autoencoder is trained, the output layer is removed and its hidden layer output is used as the input of the next autoencoder. The training procedure is repeated for all layers in the stack. When the unsupervised pre-training is completed, a further layer is added to the top of the stack and a supervised fine tuning is performed, by using a gradient descent algorithm, to finally determine the weights of the network.

The papers reported in the following, propose the use of SAEs to approach relevant open problems in SS design. Process industries historical databases are, in fact, usually data-rich but information-poor, containing mutually correlated and highly redundant variables. A wide number of methods is used, during SS design, to extract relevant information, through feature extraction procedures. Common strategies are based on cross-correlation analysis, Principal Component Analysis (PCA), Partial Least Squares Regression (PLS) and its non-linear counterpart nonlinear PLS (NLPLS), Mallow's Cp coefficient, Lipschitz's quotient, Mutual Information (MI) or Partial Mutual Information (PMI), Least Absolute Shrinkage and Selection Operator (LASSO) algorithm, and many other methods [2, 11]. In the last years, deep NNs have been proposed in the literature to this aim, due to their ability to learn hierarchical abstract feature representation from data. However, traditional deep learning algorithms feature extraction is unsupervised, i.e., the learning algorithms are not designed to extract high-level output-related features. This problem is addressed in [12], where a new variable-wise weighted SAE is proposed. Relevant input variables are chosen by using the correlation between the input of each autoencoder and the output. In the proposed method, a deep autoencoder is pre-trained, a layer at a time. The output information is incorporated by using different weights for different variables in the objective function of the autoencoder. Weights are determined by using the correlation analysis described above. A supervised final fine-tuning is then performed. An SS for a debutanizer distillation column is reported as a case study.

Further results are reported in [13], where a nonlinear variable-wise weighted SAE is proposed to learn quality-related features. The nonlinear Kendall correlations of input or feature variables with the quality variable in each autoencoder are used to obtain a corresponding weighted reconstruction objective function, which is designed to learn quality-related features, layer by layer. The same case study of [12] is considered. A hybrid variable selection method is proposed in [14], again based on SAEs. In this case, at each iteration, the ranking of the input variables is performed, taking into account the mutual information between the input variables and the prediction error. Less relevant variables are removed and the weights of the first layer of the networks are updated. Two applications are reported in the paper: the estimation of the concentration of carbon monoxide and ethylene in a gas mixture and the estimation of the rotor deformation of the air preheater in a thermal power plant boiler. SAEs are, also, proposed in [15] for the operation optimization of a coke dry quenching process and in [16] to estimate the oxygen content in flue gasses in a 1000 MW ultra-supercritical unit of a coal-fired thermal power plant. A new SS modelling method, which integrates autoencoders and supports vector regression, is proposed in [17]. In the paper, an autoencoder is used for determining a robust high-level feature representation of the data. The support vector regression model is used for the output estimation, on the basis of the obtained features. The described method is used for solving the problem described in [16].

The second class of deep learning SS design methodologies is based on the use of DBNs. This structure, built as a stack of Restricted Boltzmann Machines (RBMs), is widely described in the next section, along with its learning algorithm. Here, some relevant applications are briefly described. The paper [18], appeared in 2014, represents, as for the author knowledge, the first use of deep learning in the SS design field. In the paper, a DBN is used to design an SS, which estimates the heavy diesel 95% cut point of a crude distillation unit. The advantages introduced by the use of a deep learning strategy are widely discussed in the paper, as regards the improved representation ability, the possibility of extracting nonlinear latent variables, and of using unlabelled data, offered by the semi-supervised training phase. Also, the efficiency in dealing with massive data is outlined. In [19], authors use a DBN for predicting the oxygen content of a combustion process. The SS uses color flame images, captured by a camera. The DBN extracts nonlinear features from the images, obtaining a better description of the combustion process. A supervised fine-tuning stage is then used for obtaining the relationship between the images and the oxygen content. Two DBN-based regression models are proposed in the paper. The first is the traditional NN trained by back-propagation, the other is the support vector regression. The method is tested in a real combustion system. Other applications of the DBN structure to SS design can be found in [20–22]. These applications will be discussed in more details in the following to outline the potentialities of DBN in SS design through suitable real-world case studies.

CNNs have also been recently used for SS design. In [23] a CNN has been proposed to predict the gas phase composition of a pyrolysis reactor, by using a moving window approach. The paper [24] proposes an SS for health monitoring and structural damages detections in spacecraft, through the estimation of vibration responses from partial vibration measurements, using a CNN.

Randomization methods have also been extended to deep networks and used to approach SSs design. In this class of learning methods, the hidden weights are randomly chosen. Either the pseudoinverse or the least square method is applied for computing the weights of the last layer. In [25], the authors use both deep networks and the randomized algorithm for nonlinear system identification. The SS has a deep structure, where increased hidden layers and decreased hidden neurons, concur at improving the modeling capacity. The RBM learning algorithm is used to train the hidden weights and the randomized algorithm is used for computing the output weights. A semi-supervised deep learning model for SS development, based on deep Extreme Learning Machines (ELM), is also proposed in [26]. It can use both labeled and unlabelled data, improving the prediction performance and robustness of SSs.

An ensemble deep kernel model is used in [27] for an industrial polymerization process. Ensemble Learning methods are usually composed of two steps, the ensemble members generation, in which some predictive models are generated, and the prediction combination. Both stages can be implemented with a wide number of strategies. In the paper, the unsupervised learning stage of a DBN is used for features extraction. A kernel learning regression model is then used for estimation the nonlinear relationship linking the features to the output. A bagging-based ensemble strategy is adopted along with the deep kernel learning method. The reliability of the prediction model is, therefore, improved. A different semi-supervised ensemble method, based on a distance-to-model criterion and local models, is described in [28]. The procedure starts with an adaptive state partition approach, which uses unlabelled samples. Then, a distance-to-model criterion is introduced for selective ensemble learning. The proposed strategy allows for overcoming drawbacks of the *k*-nearest neighbour method. The metric allows for describing the relationships between query samples and local models more accurately. A particle swarm optimization is performed to compute the required parameters. A debutanizer distillation column and a sulphur recovery unit are used as benchmarks.

Genetic algorithms are used in [29] to determine the optimal structure of a deep sparse autoencoder. The prediction error is used as the fitness function. An SS for a biological wastewater treatment plant is used as a benchmark.

A deep probabilistic sequential network for SS design is proposed in [30]. The model combines unsupervised feature extraction and supervised dynamic modelling. The proposed method is based on the Gaussian–Bernoulli RBM and a Recurrent Neural Network (RNN). It is applied to the design of an SS for a CO_2 absorption column.

The state of the art described in this section, clearly shows a vivid interest in deep structures by the SS community. In addition, many different structures have been proposed, because of the needing of adapting deep learning to the specific application field. Among deep structures, in the following, the focus will be restricted to DBNs, as a deep structure capable to cope with three challenges, relevant in the data-driven soft sensor design, as it will be discussed in details in next sections.

3 SSs Design

SSs can be designed by either mechanistic modelling (physical modelling), multivariate statistics, or machine learning methods. In the industrial environment, data-driven models are widely adopted, both because the complexity of involved phenomena hinders the use of physical models and because of the availability of historical datasets. Data-driven model design involves four main steps:

a. experimental data acquisition/selection;
b. model class selection;
c. model identification;
d. validation.

Each of the steps mentioned above poses specific challenges, which are, currently, the object of research activities. A short discussion of the problems arising for each step is given in the following. As it concern the item a., two classes of problems can be listed. The first one deals with the quality of available data. In fact, experimental data are generally available in the process industry databases. Unfortunately, their quality is generally not well suited to SSs design, due to noise, collinearity, outliers, and heterogeneity. Suitable strategies for data pre-filtering and outliers removal are, therefore, required [2]. A specific issue raising in the process industry, relates to the scarcity of labelled data, as it will be discussed in Sect. 5. The second aspect concerns the selection of input variables relevant to the output estimation. When linear processes/models are of interest, the correlation analysis is a suitable tool for solving the problem. This is not the case when nonlinear processes are investigated. In such a case, many selection methods have been proposed in the literature, with somewhat contrasting pros and cons [11].

As it regards the item b., many choices are needed for selecting both the model class, e.g., linear or nonlinear, static or dynamic, time variant or invariant, just to mention the main ones. Nonlinear Autoregressive with Exogenous Inputs (NARX) models are a quite general class of models, usually adopted in the field of SS design.

Multi-Input Single Output NARX models can be represented as:

$$\begin{aligned}\hat{y}(k) = f(&y(k-1), y(k-2)\ldots, y(k-n),\\ &\boldsymbol{u}(k-1), \boldsymbol{u}(k-2), \ldots, \boldsymbol{u}(k-m)\end{aligned} \tag{1}$$

where k is an integer representing the discrete time, $\hat{y}(k)$ is the estimation of the k–th sample of the system output, $\boldsymbol{u}(k)$ is the vector of the input variables, $f(\bullet)$ is a nonlinear continuous function, n is the number of output regressors (model order), and m is the number of input regressors. If the output regressors are not used, a Nonlinear Finite Impulse Response (NFIR) model is obtained. If the model does not contain any past sample of inputs and outputs it is called a static nonlinear model. NFIR models well fit SSs design since they do not require past measured values of the system output, even though modelling the system dynamics. Static models, which are the simplest possible nonlinear model structure, can be used when the system dynamics is not relevant.

Finite time delay among inputs and outputs can characterize industrial processes. It needs to be estimated by experimental data. Deep networks can help in determining the value of finite time delay in SSs design [12–14, 41].

The item c, deals with the identification of the nonlinear function $f(\bullet)$. Statistical or machine learning methods can be used to this aim [1, 2]. More recently, deep learning has emerged as a valuable tool for identifying the function $f(\bullet)$, on the basis of the available dataset. The focus of this chapter is on the application of DBNs as an identification tool.

The item d. can be addressed using suitable indices and graphs. Among possible indices, the Correlation Coefficient (CC), between the plant real process and the corresponding estimation, and the Root Mean Square Error (RMSE) are widely adopted in the literature. Further investigations deal with the statistical characteristics of the model residual, which can be addressed by using either linear or nonlinear approaches [2].

4 Deep Belief Networks

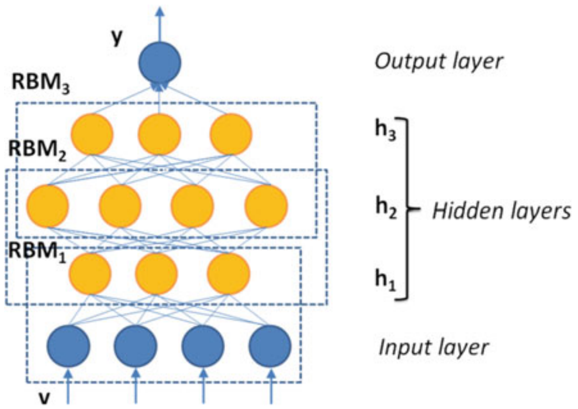
Though many deep structures have been proposed in the literature according to the problem of interest [5], this section will focus on DBNs, which will be used for the cases of study reported in the chapter.

Due to the universal approximation capability of one-hidden-layer (shallow) Multi-Layer Perceptrons (MLPs), they are commonly used in the SS design [1, 2]. The success of MLPs is due to the introduction of the back-propagation (BP) algorithm and its successive improvements for MLPs training. The BP is a supervised learning approach since it requires a suitable set of input-output pairs to be available (labelled data). Unfortunately, shallow networks revealed to be inefficient in solving complex problems [5] so that Deep Networks (DNs), i.e. networks with more than one hidden layer, were proposed as a suitable alternative. BP is not an efficient training algorithm for DNs, because it is susceptible to get stuck on poorly behaving local minima. Alternative learning approaches were, therefore, needed. This problem was solved in 2006 by Hinton and his co-workers, who introduced an algorithm capable of training DNs [4]. Many structures have been, since then, introduced to approach different classes of problems. In the following, both the structure of a DBN and its learning algorithm are described [31, 32]. A scheme of a DBN with 3 hidden layers and one output neuron, as required for MISO models, is reported in Fig. 1.

A DBN is obtained by stacking L Restricted Boltzmann Machines (RBMs), which represent the building blocks of the DN. Each RBM has two layers. The lower visible layer $\mathbf{v}_l$, $(l = 1, \ldots, L)$, representing the input, and the hidden layer $\mathbf{h}_l$, which produces the latent variables. The RBM is an unsupervised model, obtained by its parent network, i.e., the Boltzmann Machine, when connections between neurons in the same layer, either the visible or the hidden layer, are not allowed. Only connections between visible and hidden variables, therefore, exist.

In the stack, the output of each RBM is the visible layer to the next RBM. A greedy layer-wise scheme, layer by layer, is adopted for implementing the unsupervised

Fig. 1 Scheme of a three-hidden layer DBN



training of the network. In the following, the training of a generic RBM in the stack will be described. For the sake of simplicity, the subscript l will be omitted.

Unsupervised learning is performed one RBM at a time. Each RBM is trained by maximizing the joint distribution P(**v**, **h**), between its visible and hidden variables.

Such a distribution is obtained by using the energy function:

$$P(\mathbf{v},\mathbf{h}) = \frac{e^{(-Energy(\mathbf{v},\mathbf{h})}}{\sum_v \sum_h e^{(-Energy(\mathbf{v},\mathbf{h}))}} \tag{2}$$

so that the goal of the RBM training is, also, the minimization of the energy function. The denominator in (2) is a normalization factor, assuring that the summation of the joint distribution, over all possible values of v and h, is one. The energy is defined as:

$$Energy(\mathbf{v},\mathbf{h}) = -\mathbf{b}^{\mathrm{T}}\mathbf{v} - \mathbf{c}^{\mathrm{T}}\mathbf{h} - \mathbf{h}^{\mathrm{T}}\mathbf{W}\mathbf{v} \tag{3}$$

if **v** and **h** are binary vectors (binary RBMs) or as:

$$Energy(\mathbf{v},\mathbf{h}) = \sum_{\mathrm{i}} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \mathbf{c}^{\mathrm{T}}\mathbf{v} - \mathbf{h}^{\mathrm{T}}\mathbf{W}\mathbf{v} \tag{4}$$

for the case of continuous-input RBM (Gaussian RBMs). In (4) bi and $\sigma\, i$ are the mean and standard deviation of the Gaussian distribution characterizing the i-th visible input. Finally, if normalized inputs (i.e., zero mean and unity standard deviation) are considered, (4) reduces to:

$$Energy(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{v} - \mathbf{c}^{\mathrm{T}}\mathbf{v} - \mathbf{h}^{\mathrm{T}}\mathbf{W}\mathbf{v} \tag{5}$$

In (3), (4), and (5) **W, b,** and **c** are the parameters of the RBM. More specifically, **W** is the weight matrix, **b** is the bias vector of the visible layer, and **c** is the bias vector of the hidden layer. They need to be determined by a suitable learning algorithm. To this aim, the gradient descent is applied to the log-likelihood at the point **v** as follows:

$$\begin{aligned}\frac{\partial \log \sum_h \mathrm{P}(\mathbf{v},\mathbf{h})}{\partial\theta} &= \frac{\sum_h e^{-Energy(\mathbf{v},\mathbf{h})}\left(\frac{\partial[-Energy(\mathbf{v},\mathbf{h})]}{\partial\theta}\right)}{\sum_h e^{-Energy(\mathbf{v},\mathbf{h})}} - \frac{\sum_{\tilde{\mathbf{v}}}\sum_h \mathrm{e}^{-Energy(\tilde{\mathbf{v}},\mathbf{h})}\left(\frac{\partial[-Energy(\tilde{\mathbf{v}},\mathbf{h})]}{\partial\theta}\right)}{\sum_{\tilde{\mathbf{v}}h} e^{-Energy(\tilde{\mathbf{v}},\mathbf{h})}}\\ &= \sum_h P(\mathbf{h}|\mathbf{v})\left(\frac{\partial[-Energy(\mathbf{v},\mathbf{h})]}{\partial\theta}\right) - \sum_{\tilde{\mathbf{v}}} P(\tilde{\mathbf{v}},\mathbf{h})\left(\frac{\partial\left[-Energy(\tilde{\mathbf{v}},\mathbf{h})\right]}{\partial\theta}\right)\end{aligned} \tag{6}$$

where $\theta = \{\mathbf{W},\mathbf{b},\mathbf{c}\}$ contains the weight and bias matrices of the RBM.

The first term in (6) is also called the positive term. It is the conditional expectation of $\partial[-Energy(\mathbf{v},\mathbf{h})]/\partial\theta$. It is easily computed, given that the values of the conditional probability

$$P\left(h_j = 1|\mathbf{v}\right) = \frac{e^{c_j + W_j v}}{1 + e^{c_j + W_j v}} = sigm\left(c_j + W_j \mathbf{v}\right) \tag{7}$$

is known. In (7) *sigm*(.) is the usual sigmoidal function.

The second term in (6) is called the negative term. It is the expectation of $\partial[-Energy(\mathbf{v}, \mathbf{h})]/\partial\theta$ of the joint distribution P(**v**, **h**) and can become intractable for RBMs with a large number of units. It is usually approximated by using the Contrastive Divergence (CD) algorithm.

A two-stage Gibbs Sampler is used as an approximated sampling approach. A one-step Markov chain is used, usually, to this aim:

1. Sample $\mathbf{h}^1$ from $P(\mathbf{h}|\mathbf{v} = \mathbf{v}^0)$,
2. Sample $\mathbf{v}^1$ from $P(\mathbf{v}|\mathbf{h} = \mathbf{h}^1)$.

By using the approximation reported above, θ, can be easily updated [18].

After the unsupervised training, the latent variables are obtained at the topmost layer. They are further processed for estimating the DBN output. An output layer is, added to this aim. The weights of this layer are randomly initialized. The whole network is fine-tuned by using a BP-like training algorithm. The weights, obtained during the unsupervised learning phase, are used as an initial value for θ. The fine-tuning phase is the only one needing labelled data.

5 Real-World Cases Study

This section is intended to illustrate the potentialities of DBNs in SS design. Each subsection in the following refers to a challenge relevant for the SS community and describes the advantages offered by DBNs with respect to conventional NNs. Moreover, in each subsection, a real case study is used to better outline the investigated aspect. The reader interested in the cases study can find further details on the referenced literature.

5.1 Semi-supervised Learning Based Soft Sensors

As mentioned in a previous section, the scarcity of labeled data characterizes many applications and strongly limits the possibility of data-driven SS design. DBNs can offer a valuable solution to such a drawback. In fact, conventional supervised learning algorithms can use only labelled data, which are a little fraction of data generally collected in industrial databases. As a consequence, the large amount of available data can not be used for the SS design. The unsupervised learning phase of a DBN, followed by a supervised fine-tuning, perfectly matches the typical scenario of an SS design. It is usual, in fact, that physical process variables are sampled at a much faster rate than the corresponding chemical quality variables. Process variables (flows,

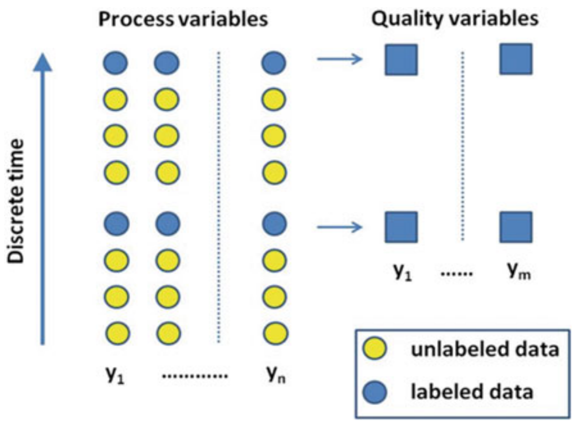


Fig. 2 Time distribution of labeled and unlabeled data in a typical database of a process industry plant

pressures, temperature, etc.) are, in fact, measured online (one sample per minute or even faster rates are used), while quality variables are sampled at a much lower sampling rate. They are, generally, measured in the laboratory, typically, once or twice a day.

A schematic of data distribution in time is reported in Fig. 2, blue-filled forms refer to input (circles) and target (squares) for labelled data, while yellow forms refer to unlabelled data. In the present case of study, labelled data correspond to lab-measured data, while unlabelled data are acquired online.

Fast sampled data (unlabelled data) can be exploited during the unsupervised training phase, while the few labelled data can be used during the supervised fine-tuning. The unsupervised training phase allows for starting the supervised phase in a better condition with respect to the randomly chosen classical BP initial conditions. This will have a beneficial effect on the prediction performance of the network [5].

Here, the advantages of using a semi-supervised learning algorithm in the design of SSs are illustrated through a real-world case of study [21]. The approach, described in the following, has been introduced in [18], where the ASTM 95% cut point index of an atmospheric distillation column has been modelled by a static model, based on DBNs. To the best of authors' knowledge, the case study described in [18] is the first reported case of application of DBNs to SSs design, as a method for exploiting unlabelled data.

In the following, the design of SSs for a Sour Water Stripping (SWS) plant of a refinery is described. Two SSs are designed for estimating the concentrations of hydrogen sulfide (H_2S) and ammonia (NH_3) in the wastewater of the SWS plant. SWS plants are used for removing pollutants in the refinery wastewater. The processed water can be either re-used by other refinery plants or released to the environment. The concentrations of H_2S and NH_3 need to be, therefore, monitored. Refineries, generally, use to this aim laboratory facilities.

A low number of samples are, therefore, available in the historical database, when compared with process data, which characterize the SWS plant history. A scheme

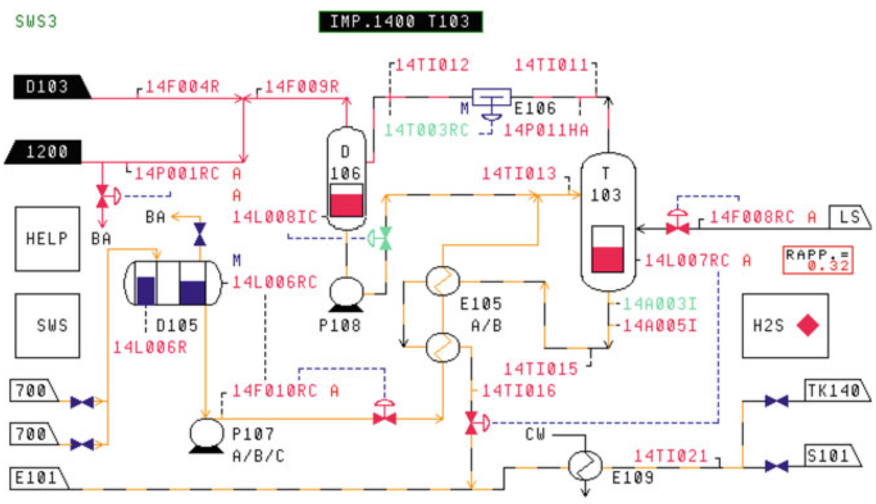


Fig. 3 Scheme of the plant (T103) and Tags of relevant variables

of the process is reported in Fig. 3. In the same figure, the relevant variable Tags are indicated.

Input variables, relevant to the SWS plant modelling, were selected with the help of plant technologists (see Table 1). Data were acquired at a sampling interval of Ts = 1 min and then averaged to obtain a value each 15 min. Data, corresponding to the time interval 2:15 a.m. to 4:30 a.m., are stored daily in the plant database, while the plant outputs are measured in the lab, once a day, at 5.00 a.m.

The SSs are two nonlinear Multi-Input Single-Output models, as reported in (1). Data were pre–filtered and normalized. A labelled dataset corresponding to 700 d was obtained. For each of them, 9 samples of the input variables are available. Unlabelled input data were, therefore, 6300. To appreciate the advantages offered by the semi-supervised approach with respect to a supervised method, the performance

Table 1 Input and output variables, tags, and units

	Variables description	Tag	Unit
u_1	Steam to T103	14F008RC	kg/h
u_2	Feed to T103	14F010RC	m^3/h
u_3	BA gas pressure	14P001RC	N/cm^2
u_4	E-106 output flow temperature	14T003RC	°C
u_5	T-103 top steam temperature	14TI011	°C
u_6	T-103 bottom to E-105 temp.	14TI015	°C
u_7	T-103 bottom from E-105 temp.	14TI016	°C
y_1	H_2S content in SWS output	1404A004	ppm
y_2	NH_3 content in SWS output	1404A124	ppm

obtained by using a DBN are compared with those of an MLP. The MLP was trained by using the Levenberg-Marquardt algorithm, randomly splitting the 700 labelled data into learning, validation, and test data sets. The number of hidden neurons of both networks was determined by a trial-and-error strategy. A growing strategy was used. Best networks were selected based on their estimation capabilities, evaluated on the validation dataset. A 7-5-1 MLP has been obtained, for H_2S estimation. A 7-7-1 MLP was obtained for NH_3 estimation. The performance of the NH_3 model is reported in Fig. 4. In Fig. 4a, the estimated and measured output are compared. In Fig. 4b, the EDA residual analysis (autocorrelation, lag plot, histogram, and normal probability plot) is shown.

The DBN was trained with the Matlab® DeeBNetV3.2 [33]. The 6300 unlabelled patterns were used for the pre-training, while the supervised fine-tuning was performed with the labelled 700 patterns. The number of layers and neurons was selected by using a trial-and-error procedure. DBNs with 2 and 3 hidden layers were taken into account. Moreover, the maximum number of neurons, for each layer, was limited to 20. Deeper networks were not taken into account to avoid too many hyper-parameters. A growing strategy was adopted also in this case.

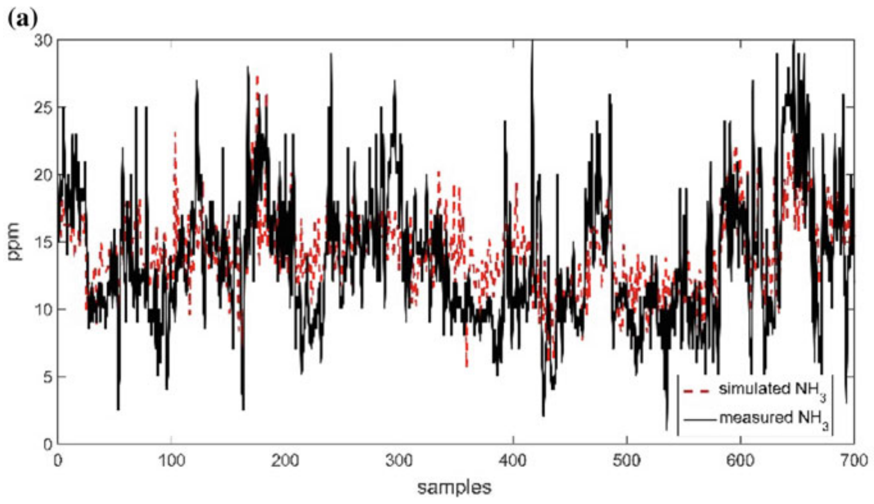
A 7-20-11-3-1 DBN was selected for the H_2S estimation; a 7-5-6-8-1 network was selected for the NH_3. Figure 5 reports the results obtained for the estimation of the NH_3. In Table 2, RMSE and CC are reported, along with their improvements with respect to MLP based SSs.

The residual analysis shows that the autocorrelation plot of the residuals is inside the 95% confidence band for smaller time lags than those obtained by using MLP based SSs. Also, RMSE and CC are improved. Results reported above give evidence that that DBNs can offer advantages over simpler strategies, like BP, when a large number of unsupervised data is available and the model performance are affected by scarcity of labelled data.

5.2 Reducing Model Complexity by Using DBN

The following case studies are intended to focus on the potentialities offered by DBNs in reducing the model complexity, both as regards the model order and the number of model parameters. This is due to the fact that using multiple layers of representation allows for approximating a more complex function, with a lower number of parameters. Two case studies are illustrated. The first refers to the design of an SS, required to estimate the Research Octane Number (RON) for a Reformer Unit in a refinery [22, 34]. In this case, the use of a DBN allowed obtaining a dynamic FIR model with a lower number of parameters. The second case of study refers to the design of a Low-order Nonlinear FIR SS for Ionic Electroactive actuators [20]. In this case, the model capability offered by the DBN allowed for obtaining a low-order model, i.e. a model with a lower number of regressors.

In the first case, the RON value is measured by a Near-InfraRed (NIR) analyser. The SS is required for estimating the RON value during maintenance. NFIR models



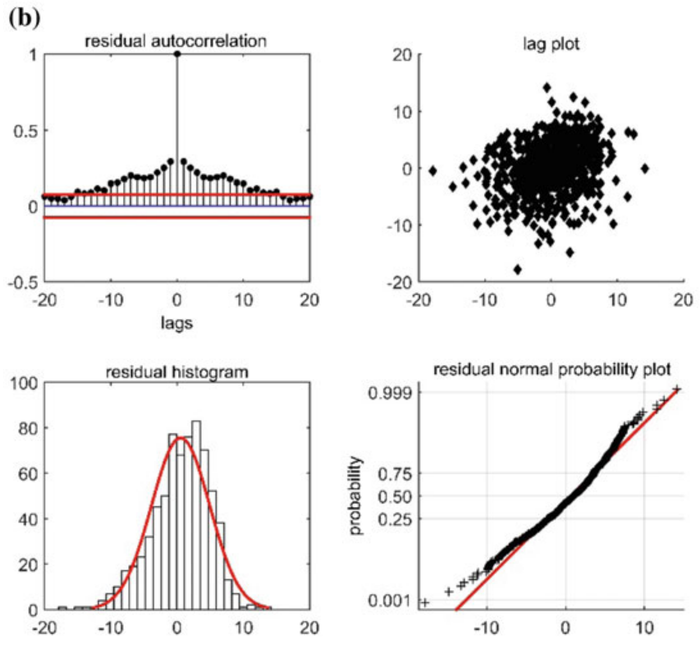


Fig. 4 Performance of the MLP model for the NH_3 estimation. **a** Time plot of the SS estimation and the measured values; **b** EDA analysis of the model residual

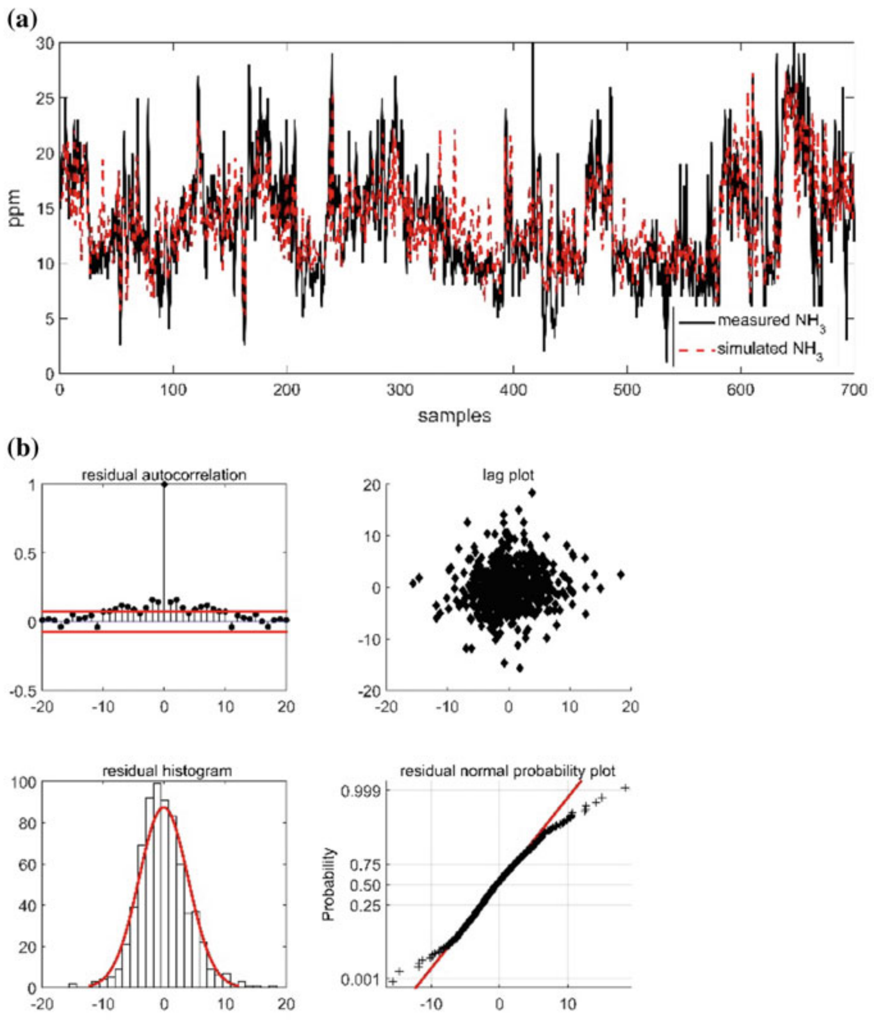


Fig. 5 Performance of the DBM for the NH_3 estimation. **a** Time plot of the SS estimation and the measured values; **b** EDA analysis of the model residual

Table 2 Model performance

	RMSE	CC	ΔRMSE%	ΔCC%
H_2S_MLP	1.33	0.59		
NH_3_MLP	4.38	0.57		
H_2S_DBM	1.20	0.67	−9.8	13,6
NH_3_DBM	4.09	0.64	−6.6	12,3

should, therefore, be considered. Data come from the historical database of the refinery. In the process, petroleum distillate (naphtha) reacts with a catalyst. The process upgrades the low-RON feed into a high-RON liquid product. The plant works in two different conditions, depending on the production policy.

In a first study [42], several modelling strategies were investigated for the plant. In that contribution, it was shown that the process is nonlinear and simple modelling structures do not give satisfactory performance. Deep structures were, therefore, proposed. The structures consisted of two levels. The first level contains two models, each designed for a single working condition. The second level fuses the output of the first level models. The first level implements either linear or nonlinear dynamic models, by using shallow MLPs. The second one uses a fuzzy rule for combining the first-level outputs.

In this subsection, it is shown that DBNs can be used as an alternative method, for solving the described application. The use of DBNs gives a simpler designs procedure and uses a lower number of parameters. Eventually, improvements in the model performance are obtained. A scheme of the plant is reported in Fig. 6.

The RON specification for the Unit is daily established. More specifically, the maximum capacity of the plant is 250 m^3/h of naphtha flow. When it is fed at its maximum capacity, the RON target value is 96.5. When the flow is 220 m^3/h the RON target value is raised to 99. Data have been selected from the refinery database. The sampling time is $T_s = 3$ min. The SS inputs are the temperatures of the four reactors, RX 101 to RX 104 (in_1 to in_4), the total feed (in_5), and the pressure of the Debutanizer plant T104 head feed (in_6). After outliers filtering, 9131 samples were available for the SS design. In this case, no unlabelled data are available, and both the unsupervised training phase and the fine-tuning use the same dataset. Regressors

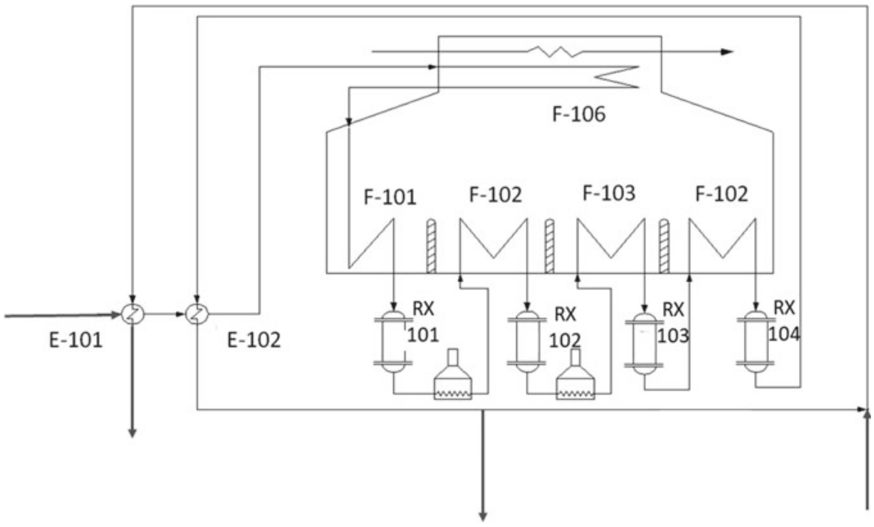


Fig. 6 The powerformer unit

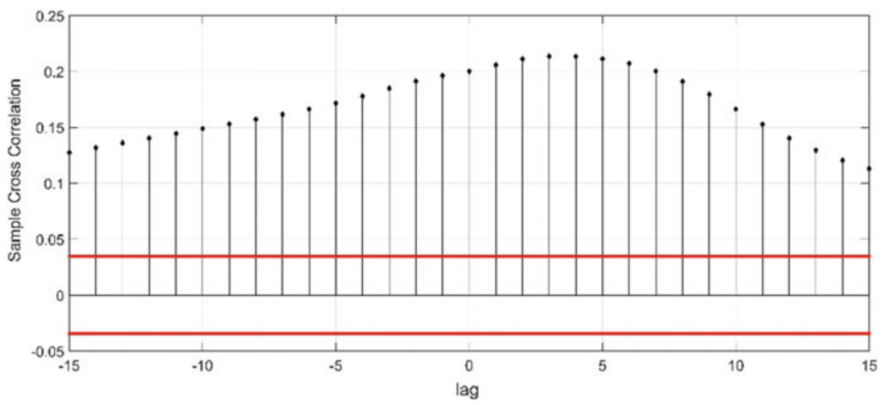


Fig. 7 Cross-correlation between *in₁* and *y*

of the model were selected based on the correlation analysis, among the RON value and the inputs, in_1 to in_6. As an example, in Fig. 7 the cross-correlation between in_1 and y is reported. It indicates that the regressor at lag 4 is more relevant.

Based on similar considerations, the following model has been used:

$$y(k) = f(in_1(k-4), in_2(k-4), in_3(k-4), in_4(k-4), in_5(k), in_6(k)) \quad (8)$$

Three-hidden-layer DBNs were considered. An exhaustive search strategy for selecting the number of neurons was adopted. The maximum number of neurons for each layer was fixed to 20. The best working network was a DBN with 8, 12, and 12 neurons in the three hidden layers. The obtained results are reported in Table 3. Time plots of the SS estimation, of the residual, and the corresponding histogram, are reported in Fig. 8.

To appreciate the improvements obtained by the DBN, with respect to other deep structures in [42], a comparison of the value of the CC, obtained for the validation dataset, is reported in the following. Structures investigated in [42] are:

- one linear model, working on the whole data set;
- two linear models, activated by using a fuzzy selection algorithm;

Table 3 The DBN model performance (Network 6-8-12-12-1)

# Of parameters	332
RMSE (Training)	0.240
RMSE (Testing)	0.257
RMSE (Validation)	0.280
CC (Training)	0.971
CC (Testing)	0.965
CC (Validation)	0.958

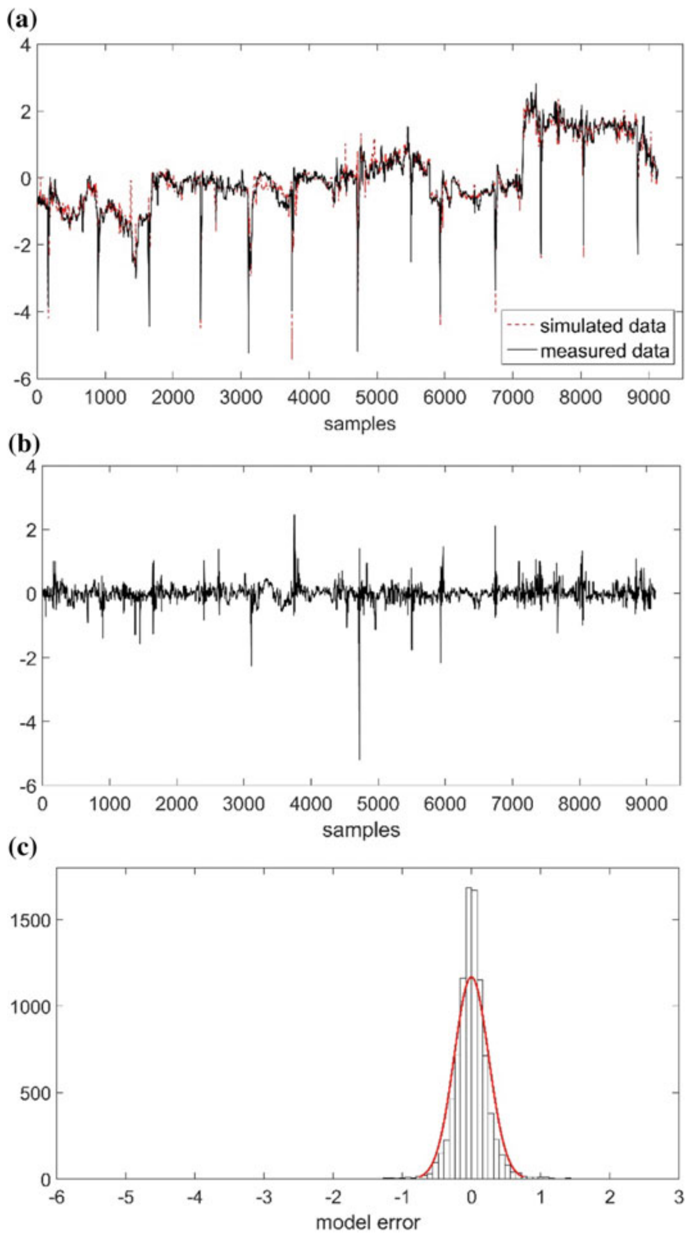


Fig. 8 Time plot of the model estimation (**a**), model residual (**b**), and residual histogram (**c**)

Table 4 CC comparison for models estimating the RON value

Model structure	CC
One linear model working on the whole data set	0.7742
Two linear models activated by using a fuzzy selection algorithm	0.7977
One shallow neural model working on the whole data set	0.8541
Two shallow neural models activated by using a fuzzy selection algorithm	0.8722
DBN model	**0.958**

- one neural model, working on the whole data set;
- two neural models, activated by using a fuzzy selection algorithm.

The values of CC for all the mentioned models are reported in Table 4. As a general trend, it is possible observing that deep structures outperform shallow structures. Such a tendency is respected both for linear and nonlinear structures. Moreover, the DBN works much better than all other tested structures. Finally, the design procedure for DBN is much simpler than that required for the deep structures proposed in [42]. This further shows the advantages obtained when DBNs are used in SSs design.

As the second case of study, the possibility of obtaining a low-order model for an electromechanical transducer is described. Polymeric transducers have raised the interest of both the scientific and industrial community since they are a bundle of enabling technologies, towards the realization of Smart Systems. Among polymeric composites, capable of electromechanical transduction, Ionic Polymer-Metal Composites (IPMCs) have raised a vivid interest with many scientific studies published in the last three decades [35].

In the following, the application of deep networks for the design of a low-order SS, modeling the vibrations of an IPMC actuator, in a cantilever configuration, is described [20]. Part of the interest in IPMCs is due to the possibility of using them in medical applications [36], where both the complexity and the size of the actuating system are relevant. An SS can, therefore, help in limiting the complexity of the system by eliminating the need for a hardware sensor for the independent measurement of the transducer actual position. Moreover, low-order models limit the computational load, with beneficial effects in real-time applications. To this aim, a strategy for lower-order NFIR model design was investigated, among linear models, PCA, shallow and deep neural networks.

IPMCs consist of a bulk ionomeric polymer, such as Nafion®, covered on both faces by two electrodes. These are, generally, realized by using noble metals. In the bulk of the composite, fixed anions and mobile cations are in electric equilibrium. Mobile cations can, anyway, migrate when a voltage signal (of the order of few volts) is applied to the electrodes. The charge unbalance is the origin of the IPMC bending.

Data were acquired by measurement surveys, performed in an environment with controlled temperature and humidity. The IPMC actuator was mounted as shown in Fig. 9. A sine sweep voltage, with frequency in the range 50 MHz–50 Hz, in 10 s, and a peak-to-peak amplitude of 5 V, was used as the electric input.

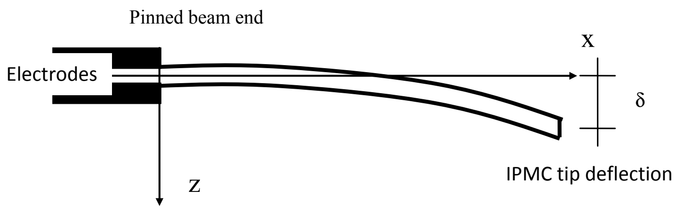


Fig. 9 The IPMC actuator in the cantilever configuration

Based on the NFIR model (1), the simplest model structure was searched for. Different model classes and orders were, therefore, investigated. In details, according to gray box models already available in the literature [37, 38], model orders in the range 9 to 3 were identified for the following model structures:

(1) *Linear dynamic FIR models*
(2) *NFIR models, based on one hidden layer MLPs*
(3) *NFIR models based on Nonlinear Principal Component Regression (NPCR)*
(4) *NFIR models, based on two–hidden–layers MLPs*
(5) *NFIR models, based on three hidden layers DBNs*
(6) *Three hidden layers DBNs with regressors k–9, k–7, k–5, k–3 and k*
(7) *Three hidden layers DBNs with regressors k–9, k–6, k–3, and k.*

Models (1) to (7) have been compared by estimating the RMSE of the model residual and the CC, between the estimated and measured output. The coefficients have been computed both on the learning and test datasets. Obtained results are reported in Tables 5a, 5b, 6a and 6b, respectively. The analysis of the reported tables shows that deeper structures work better than shallower ones. Moreover, the higher the order model, the better the performance. The best SS has been obtained with a DBN, implementing a 9-th order NFIR. The network structure consists of 10, 11, and 6 neurons in the hidden layers. Moreover, the DBN based 5-th order model (9, 5, and 5 neurons in the hidden layers), works better than any shallow structure considered. The same network works better than the best performing two-hidden-layer model, i.e., the 7-th order model. MLPs, trained with BP, suffer from overfitting problems.

On the contrary, DBNs looks not to be affected by overfitting, as shown by the agreement between the learning and test performance. The reported results give evidence that the use of DBNs allows for obtaining better working low-order models of the IPMC actuator. It is worth observing that the 9-th order DBN model has 300 parameters, while the 5-th order DBN requires only 122, which is further evidence of the model simplification obtained by using DBNs. Such a simplification represents a valuable result when online applications are of interest since both the lower order and the lower number of parameters reduce the computational complexity. In Fig. 10, the time plot of actual IPMC tip deflection is compared with the 5-th order DBN based SS estimation.

Table 5 A. Values of CC for different model structures and model orders (learning data). B. Values of CC for different model structures and model orders (test data)

Model structure	Model order						
	9	8	7	6	5	4	3
(A)							
1	0.46	0.44	0.43	0.42	0.41	0.38	0.35
2	0.98	0.98	0.95	0.95	0.92	0.92	0.87
3	0.96						
4	0.995	0.996	0.995	0.993	0.994	0.981	0.979
5	0.997	0.997	0.996	0.995	0.991	0.990	0.976
6	0.996						
7	0.996						
(B)							
1	**0.44**	0.39	0.32	0.26	0.21	0.15	0.12
2	0.69	0.72	**0.87**	0.78	0.69	0.50	0.29
3	**0.84**						
4	0.901	0.920	**0.956**	0.838	0.670	0.568	0.620
5	**0.987**	0.985	0.976	0.981	0.975	0.955	0.720
6	**0.981**						
7	**0.981**						

Though the cases of study described in this subsection refer to quite different phenomena, reported results show the potentialities of DBN in reducing both the complexity of the structure and of the design structure of SS.

5.3 *Gaining Insight on the SS Structure by Exploiting DBN Features*

Results reported in the literature show the capability of deep networks to extract nonlinear features from data, going from low-level features to higher-level ones. In this section, a case study is reported about the possibility of using features extracted by a DBN for inferring information on the model structure. More specifically, features extracted during the unsupervised learning phase of DBNs are used to estimate the unknown value of the measurement finite time delay that affects an industrial plant. As will be described in the following, the delay estimation was obtained by a cross-correlation analysis between the DBN features and the recorded plant output.

The approach was applied for the design of an SS for estimating the butane percentage in the penthane bottom flow (C4 in C5) of a debutanizer distillation

Table 6 A. Values of RMSE for different model structures and model orders (learning data). B. Values of RMSE for different model structures and model orders (test data)

Model structure	Model order						
	9	8	7	6	5	4	3
(A)							
1	0.79	0.80	0.80	0.81	0.81	1.25	1.83
2	0.20	0.17	0.27	0.27	0.34	0.94	0.44
3	0.25						
4	0.087	0.078	0.093	0.104	0.099	0.650	0.183
5	0.073	0.071	0.078	0.085	0.119	0.297	0.192
6	0.076						
7	0.074						
(B)							
1	**1.15**	1.18	1.21	1.23	1.25	1.25	1.27
2	0.94	0.98	0.89	**0.86**	0.94	0.94	1.33
3	**0.85**						
4	0.560	0.514	**0.377**	0.738	0.650	0.650	1.296
5	**0.222**	0.226	0.286	0.266	0.297	0.297	0.888
6	**0.274**						
7	**0.253**						

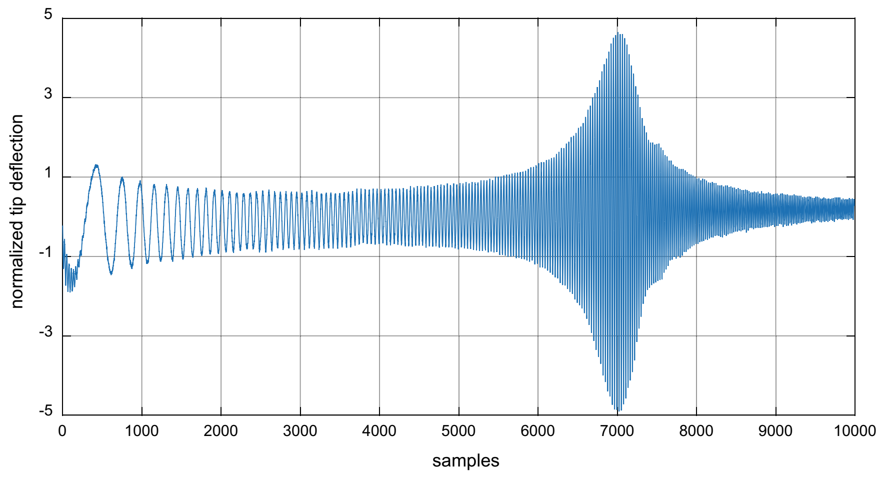


Fig. 10 Time plots of the actual IPMC output *vs*. its estimation

column. Data comes from a refinery working in Sicily, Italy. The debutanizer column represents a case of study widely considered in the SSs literature [12, 13, 28, 39, 40]. A key aspect making this an interesting case of study is that the plant is affected by a large unknown measurement delay. The delay is caused by the position of the measurement hardware, used to evaluate the quality of the process output. In fact, relevant quantities are measured by using an analyzer, which is settled in another column. A view of the relative position of the SS and the corresponding analyzer is reported in Fig. 11, where the encircled A refers to the position of the measurement hardware.

Table 7 reports the input variables considered in the SS design, as suggested by the plant technologists. The reader interested in further details on the measured quantities can refer to [41].

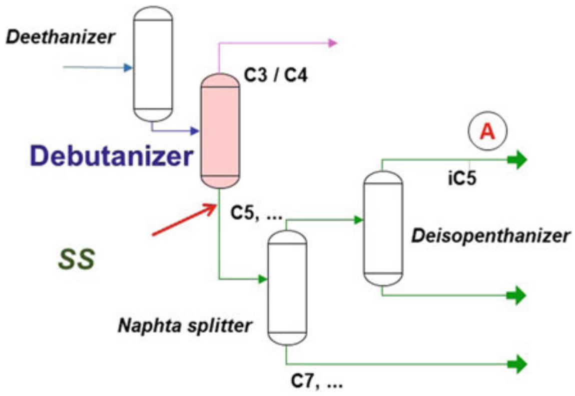


Fig. 11 A scheme of the involved columns

Table 7 Input variables

Variable name	Variable description
u_1	Head flow temperature
u_2	Head flow pressure
u_3	Head reflux charge
u_4	PLG Flow to plant 900
u_5	Tray 6 temperature
$u_{6\text{-}7}$	Mean value between temperature of vapors from E108A and E108B
u_8	Vacuum bottom valve
u_9	Temperature of flow from E150B
u_{10}	Temperature of flow from E150A
u_{11}	Reboilers residual flow
u_{12}	Bottom flow from E105

In [40], an SS was obtained by using a complex neural structure. More specifically, the three MLPs were cascaded for implementing a NARX model. In the following DBNs are exploited for designing a simplified SS structure, implementing an NFIR model [41].

Based on the results of the cross-correlation analysis between the output and candidate inputs the following model structure was obtained:

$$\begin{aligned}\hat{y}(k) = & f(u_1(k-10), u_2(k-10), u_3(k-10), u_3(k-12), u_3(k-14), u_3(k-16), u_3(k-18), u_4(k-10),\\ & u_5(k-10), u_5(k-12), u_5(k-14), u_5(k-16), u_5(k-18),\\ & u_{6-7}(k-10), u_{6-7}(k-12), u_{6-7}(k-14), u_{6-7}(k-16), u_{6-7}(k-18),\\ & u_8(k-10), u_9(k-10), u_{10}(k-10), u_{11}(k-10), u_{12}(k-10),\end{aligned} \tag{9}$$

Time lags in (9) take into account the unknown finite delay. A solution to this problem, exploiting DBNs features is described in the following. The proposed procedure can be used to approach the general problem of unknown finite delay estimation in SSs design, when nonlinear processes are of interests and/or ad hoc experiments cannot be executed. In fact, in the process industry, SSs are generally designed based on data stored in the historical plant databases, which are recorded during the normal working conditions. Signals could be, therefore, not optimal for the model identification. Selecting the model order and finite delay can be therefore a difficult task.

In the following, it is described how features, obtained after the unsupervised learning phase of a DBN, can give an alternative representation of inputs, useful when searching for the value of the finite delay. Let us indicate with *max_lag* the maximum measurement time delay. Insight about its value could be given by plant technologists. Let us consider with DBN with m neurons in the last hidden layer, i.e., m latent variables, and let us indicate with L a matrix containing the m features:

$$L = \left[\boldsymbol{l}_1 \ldots \boldsymbol{l}_m \right] \tag{10}$$

where $\boldsymbol{l}_i \in \Re^{\mathrm{N}}$ is the i-th latent variable and N is the training set size. Collect the time-shifted values of the output in $\boldsymbol{y}_{lag}$, as it follows:

$$\boldsymbol{y}_{lag} = \left[\boldsymbol{y}_0 \cdots \boldsymbol{y}_i \cdots \boldsymbol{y}_{\max\_lag} \right] = \begin{bmatrix} y(1) & \ldots & y(\max\_lag + 1) \\ \ldots & \ldots & \ldots \\ y(N - \max\_lag) & \ldots & y(N) \end{bmatrix}. \tag{11}$$

CC values are collected in a vector $\boldsymbol{CCl}_i \in \Re^m$. It contains the CC between the i-th column of $\boldsymbol{y}_{lag}$ and each latent variable. The norms of vectors $\boldsymbol{CCl}_i$ can be computed as:

$$d_i = \|\boldsymbol{cl}_i \cdot \boldsymbol{cl}_i^T\| \tag{12}$$

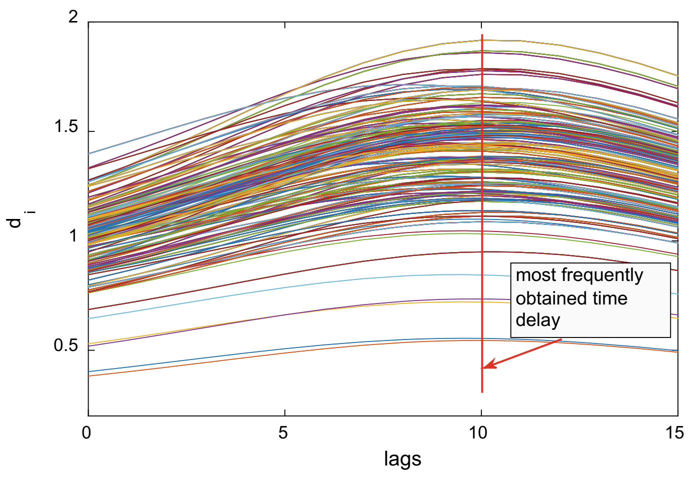


Fig. 12 Plots of the d_i coefficients *vs*. the number of lags

where $i \in [0, max\_lag]$. The value i that maximizes (12) is assumed as the system delay:

$$delay = arg_i max(d_i). \tag{13}$$

The proposed procedure revealed to be robust with respect to the DBN structure, as shown in Fig. 12. It reports d_i as a function of the time delay, for a number of DBNs. It can be observed that most of the curves have a maximum for a delay equal to 10, this value was assumed as the measurement delay to be used in the SS design.

Having estimated the finite delay, the supervised fine-tuning learning of DBNs was possible. SSs performance was compared by estimating the RMSE, between the plant output and the corresponding model estimation, and the CC, between the same quantities. The DBN with 10-6-6 hidden neurons gave the best results, which are summarized in Table 8, for the learning, validation, and test data sets, respectively.

A comparison between the plant output and the SS estimation is reported in Fig. 13.

The methodology described in this subsection, can not be applied unless a tool for unsupervised feature extraction is available. Using DNBs revealed a promising approach since DBNs allow for both feature extraction and output estimations, after the fine tuning of the network is performed, in a simple and efficient way.

Table 8 Perfomance of the DBN SS

	Learning	Validation	Test
CC	0.93	0.81	0.74
RMSE	0.88	2.42	2.07

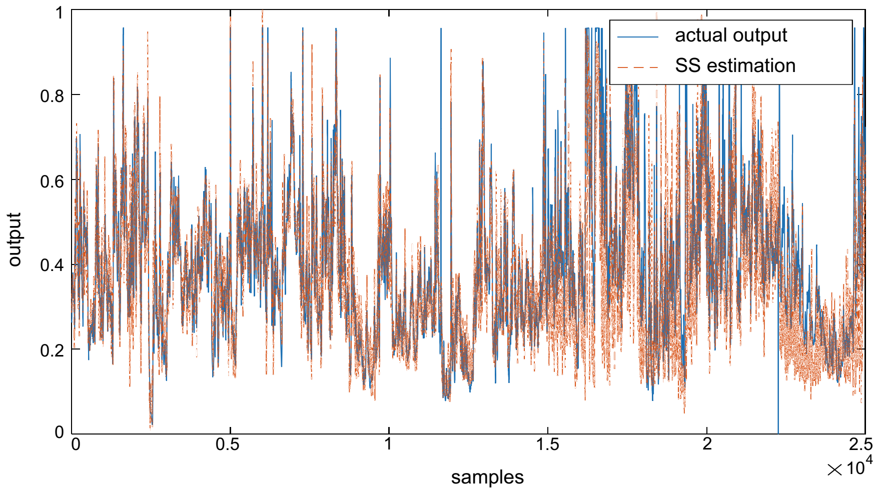


Fig. 13 Time plots of the actual debutanizer output *vs*. its estimation

6 Conclusion

Deep learning has been widely and successfully applied in many application fields, to the point that it represents the state of the art in solving a large number of real-world problems. Only recently, the promising capabilities of deep learning have been exploited in the design of SSs. As a matter of fact, there are peculiarities in the deep learning that can face unsolved problems in this field, especially if SSs for the industry are considered. This chapter focused on the application of DBNs for SS design and evidence was given, by real-world cases study, of how these structures can help in learning more complex models, reducing computational complexity, efficiently use unlabelled data, extracting high-level features from data and exploiting them in SS design. Further deep structures, such as SAE, CNN, ensemble methods, randomized methods, have been, also, proposed in the literature and successful results have been described. These deep structures and their performance in SS design, have been described in the state of the art section of this chapter.

The full exploitation of deep learning in SSs design requires that some open problems are solved. One of the problems is the increased number of hyperparameters, which is a direct consequence of the increased number of used layers. Further, theories and metrics about the role of the semi-supervised phase in the SS performances are to be introduced. Also, features extraction and interpretability should be further investigated. In fact, though high-level features are extracted from input data, their relevance to the plant output is not taken into account, nor any physical meaning can be attributed to the extracted features. Even methods for determining the number of features relevant for the SS are to be investigated.

As a final remark, it is worth noticing that deep learning has been introduced for solving problems in the presence of big datasets. This is, generally, not the case in SS design. On the contrary, the designer has to face the problem of historic data scarcity. The designer needs to carefully consider such a difference and adapt the network structure to the available dataset. Notwithstanding the open problems mentioned above, promising results obtained by applying deep learning to SS design, including the ones based on DBNs described in this contribution, allows foreseeing that they can assume a relevant role in this field and produce state of the art SSs.

References

1. Kadlec, P., Gabrys, G., Strandt, S.: Data-driven Soft sensors in the process industry. Comp. Chem. Eng **33**, 795–814 (2009)
2. Fortuna, L., Graziani, S., Rizzo, A., Xibilia, M.G.: Soft sensors for monitoring and control of industrial processes. Advances in Industrial Control, Springer (2006)
3. Ge, Z., Huang, B., Song, Z.: Nonlinear semi-supervised principal component regression for soft sensor modeling and its mixture form. J. Chemom. **28**, 793–804 (2014)
4. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
6. Hinton, G.E.: Learning multiple layers of representation. Trends Cognit. Sci. **11**, 428–434 (2007)
7. Yu, J., Hong, C., Rui, Y., Tao, D.: Multi-task autoencoder model for recovering human poses. IEEE Trans. Ind. Electron. **65**(6), 5060–5068 (2018)
8. Lee, S., Chang, J.H.: Oscillometric blood pressure estimation based on deep learning. IEEE Trans. Ind. Inf. **13**(2), 461–472 (2017)
9. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Muller, K.R.: Evaluating the visualization of what a deep neural network has learned. IEEE Trans. Neural Netw. Learn. Syst. **28**(11), 2660–2673 (2017)
10. Ren, S., Girshick, R., Girshick, R, Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017)
11. Xibilia, M.G., Gemelli, N., Consolo, G.: Input variables selection criteria for data-driven soft sensors design. In: Proceedings of 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC) 16–18 May 2017, Lamezia Terme, Italy (2017)
12. Yuan, X., Huang, B., Wang, Y., Yang, C., Gui, W.: Deep learning-based feature representation and its application for soft sensor modeling with variable-wise weighted SAE. IEEE Trans. Ind. Inf., in press
13. Yuan, X., Ou, C., Wang, Y., Yang, C..: Nonlinear VW-SAE based deep learning for quality-related feature learning and soft sensor modeling. In: Proceedings of IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society, 21–23 Oct. (2018)
14. Wang, X., Liu, H.: A new input variable selection method for soft sensor based on stacked auto-encoders. In: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), December 12–15, 2017, Melbourne, Australia (2017)
15. Wang, J.G., Zhao, J.H., Shen, T., Ma, S.W., Yao, Y., Chen, T., Shen, B., Wu, Y.: Deep learning-based soft-sensing method for operation optimization of coke dry quenching process. In: Proceedings of the 35th Chinese Control Conference July 27–29, 2016, Chengdu, China (2016)
16. Yan, W., Tang, D., Lin, Y.: A data-driven soft sensor modeling method based on deep learning and its application. IEEE. Trans. Ind. Electron. **64**(5), 4237–4245 (2017)
17. Lin, Y., Yan, W.: Study of soft sensor modeling based on deep learning. In: Proceedings of the 2015 American Control Conference, July 1–3, 2015. Chicago, IL, USA (2015)
18. Shang, C., Yang, F., Huang, D., Lyu, W.: Data-driven soft sensor development based on deep learning technique. J. Process Control **24**(3), 223–233 (2014)

19. Liu, Y., Fan, Y., Chen, J.: Flame images for oxygen content prediction of combustion systems using DBN. Energy Fuels **31**, 8776–8783 (2017)
20. Andò, B., Graziani, S., Xibilia, M.G.: Low-order nonlinear finite impulse response soft sensors for ionic electroactive actuators based on deep learning. IEEE Trans. Instrum. Meas. **68**(5), 1637–1646 (2019)
21. Graziani, S., Xibilia, M.G.: A deep learning based soft sensor for a sour water stripping plant. In: Proceedings of IEEE I2MTC 2017, Torino, Italy, 22–25 May (2018)
22. Graziani, S., Xibilia, M.G.: Deep Structures for a reformer unit soft sensor. In: Proceedings of INDIN 2018, IEEE 16th International Conference on Industrial Informatics, Porto, Portugal, 18–20 July (2018)
23. Zhu, W., Ma, Y., Zhou, Y., Benton, M., Romagnoli, J.: Deep learning based soft sensor and its application on a pyrolysis reactor for compositions predictions of gas phase components. Comput. Aided Chem. Eng. **44**, 2245–2250 (2018)
24. Sun, S.B., He, Y.Y., Zhou, S.D., Yue, Z.J.: A data-driven response virtual sensor technique with partial vibration measurements using convolutional neural network. Sensors **17**, 12 (2017)
25. De la Rosa, E., Yu, W.: Randomized algorithms for nonlinear system identification with deep learning modification. Inf. Sci. **364–365**, 197–212 (2016)
26. Yao, L., Ge, Z.: Deep learning of semi-supervised process data with hierarchical extreme learning machine and soft sensor application. IEEE. Trans. Ind. Electron. **65**(2), 1490–1498 (2018)
27. Liu, Y., Yang, C., Gao, Z., Yao, Y.: Ensemble deep kernel learning with application to quality prediction in industrial polymerization processes. Chemom. Intell. Lab. Syst. **174**, 15–21 (2018)
28. Shao, W., Tian, X.: Semi-supervised selective ensemble learning based on distance to model for nonlinear soft sensor development. Neurocomputing **222**, 15–21 (2017)
29. Qiu, Y., Liu, Y., Huang, D.: Data-driven soft-sensor design for biological wastewater treatment using deep neural networks and genetic algorithms. J. Chem. Eng. Jpn. **49**(10), 925–936 (2016)
30. Sun, Q., Ge, Z.: Probabilistic sequential network for deep learning of complex process data and soft sensor application. IEEE Trans. Ind. Inf., in press
31. Fischer, A., Igel, C.: Training restricted Boltzmann machines: an introduction. Pattern Recogn. **47**(1), 25–39 (2014)
32. Deng, L., Yu, D.: Deep learning: methods and applications. Found. Trends Signal Process. **7**(3–4), 197–387 (2013)
33. Keyvanrad M.A., Homayounpour, M.M.: A brief survey on deep belief networks and introducing a new object-oriented toolbox (DeeBNet). arXiv:1408.3264 [cs] (2014)
34. Fortuna, L., Giannone, P., Graziani, S., Xibilia, M.G.: Virtual instruments based on stacked neural networks to improve product quality monitoring in a refinery. IEEE Trans. Instrum. Meas. **56**, 95–101 (2007)
35. De Luca, V., Digiamberardino, P., Di Pasquale, G., Graziani, S., Pollicino, A., Umana, E., Xibilia, M.G.: Ionic electroactive polymer metal composites: fabricating, modeling, and applications of postsilicon smart devices. J. Polym. Sci., Part B: Polym. Phys. **51**, 699–734 (2013)
36. Shahinpoor, M., Kim, K. J.: Ionic polymer–metal composites: IV. Industrial and medical applications. Smart Mater. Struct. **14**, 197 (2005)
37. Newbury, K.M., Leo, D.J.: Linear electromechanical model of ionic polymer transducers–Part II: experimental validation. J. Intel. Mat. Sys. Struct. **14**, 343–358 (2003)
38. Bonomo, C., Fortuna, L., Giannone, P., Graziani, S., Strazzeri, S.: A nonlinear model for ionic polymer metal composites as actuators. Smart Mat. Struct. **16**, 1–12 (2007)
39. Fortuna, L., Graziani, S., Xibilia, M.G.: Virtual instruments in refineries. IEEE Instrum. Meas. Mag. **8**, 26–34 (2005)
40. Fortuna, L., Graziani, S., Xibilia, M.G.: Soft sensors for product quality monitoring in debutanizer distillation columns. Control Eng. Pract. **13**, 499–508 (2005)

41. Graziani, S., Xibilia, M.G.: Design of a soft sensor for an industrial plant with unknown delay by using deep learning. In: Proceedings IEEE I2MTC 2019, Auckland, New Zealand, 20–23 May (2019)
42. Fortuna, L., Graziani, S., Xibilia, M.G., Barbalace, N.: Fuzzy activated neural models for product quality monitoring in refineries. IFAC Proc. Volumes **16**, 159–164 (2005)

Case Study: Deep Convolutional Networks in Healthcare

Mutlu Avci, Mehmet Sarıgül and Buse Melis Ozyildirim

Abstract Technological improvements lead big data producing, processing and storing systems. These systems must contain extraordinary capabilities to overcome complexity of the big data. Therefore, the methodologies utilized for data analysis have been evolved due to the increase in importance of extracting information from big data. Healthcare systems are important systems dealing with big data analysis. Deep learning is the most applied data analysis method. It becomes one of the most popular and up-to-date artificial neural network types with deep representation ability. Another powerful ability of deep learning is providing feature learning through convolutional neural networks. Deep learning has wide implementation areas in medical applications from diagnosis to treatment. Various deep learning methods are applied to the biomedical problems. In many applications, deep learning solutions are modified in accordance with the requirements of the problems. Through this chapter the most popular and up-to-date deep learning solutions to biomedical problems are discussed. Studies are analyzed according to problem characteristic, importance of solution, requirements and deep learning approaches to solve them. Since the deep learning systems have very effective image and pattern recognition ability, biomedical imaging becomes one of the most suitable application areas. During the first diagnosis and continuous tracking phase of the patients, deep learning systems offer very effective aids to the medicine. Although organ, disease or data type classifications are possible for biomedical application categorization, organ and disease combination are taken into consideration in the chapter.

M. Avci
Faculty of Engineering, Biomedical Engineering Department, Cukurova University, Adana, Turkey
e-mail: mavci@cu.edu.tr

M. Sarıgül
Computer Engineering Department, Iskenderun Technical University, Hatay, Turkey
e-mail: mehmet.sarigul@iste.edu.tr

B. M. Ozyildirim (✉)
Faculty of Engineering, Computer Engineering Department, Cukurova University, Adana, Turkey
e-mail: mozyildirim@cu.edu.tr; melis.ozyildirim@gmail.com

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_3

Keywords Deep learning · Healthcare · Diagnosis systems · Machine learning

1 Introduction

Ever increasing effectivity of Convolutional Neural Network (CNN) make it one of the most popular and up-to-date artificial intelligence approach. It also has wide implementation area in medical applications from diagnosis to treatment. Various deep learning methods either directly applied or applied after modifications to the biomedical problems. In many applications, deep learning solutions are modified in accordance with the requirements of the problems. As the number of applications increases, some review studies were published in the literature [1, 2]. In [1], deep learning methods applied on various kind of data such as clinical images, electronic health records, genomics are analyzed, and challenges of deep learning methods are discussed briefly. On the other hand, in [2] deep learning architectures and their applications in medical imaging are analyzed. Through this chapter, the most popular and up-to-date deep learning solutions to biomedical problems are mentioned unlike the reviews. Deep learning solutions are analyzed according to problem characteristic, importance of solution, requirements and deep learning approaches to solve biomedical problems. Since the deep learning systems have very effective image and pattern recognition ability, biomedical imaging becomes one of the most suitable application areas. During the first diagnosis and continuous tracking phase of the patients, deep learning systems offer very effective aids to the medicine. Although organ, disease or data type classifications are possible for biomedical application categorization, organ and disease combination are taken into consideration in the chapter.

Brain diseases are one of the most popular deep learning application topics. Alzheimer's Disease and Parkinson are up-to-date diseases to be detected as earlier as possible. Early detection of them is crucial for early treatment and to prevent brain tissue damaging. Alzheimer's disease classification of clinical data for medical conditions has always been challenging, and one of the most problematic aspect. One of this kind works is given in [3] where an improved form of Deep Learning is applied to the Alzheimer's Disease Neuroimaging data. The difference of this proposed novel method for learning the manifold of 3D brain images is the absence of predefined similarity measure or a prebuilt proximity graph for the manifold space and also it may not be locally linear. In another work, a CNN is trained to distinguish an Alzheimer's brain from a normal, healthy brain [4]. The importance of this approach lies on the potential to develop a predictive model or system in order to recognize the symptoms of Alzheimer's disease. Also, stage estimation of the disease is successfully done by the CNN. Utilizing the CNN, functional magnetic resonance images (MRI) data of Alzheimer's subjects are separated with 96.85% accuracy on test data. In some cases, combining multi-modality brain data for disease diagnosis commonly leads to performance improvement [5]. Since the data may be incomplete, some modality might be missing for some subjects. A deep learning-based framework for estimating multi-modality imaging data is proposed in [5]. The CNNs in this work has volumetric input and output modalities. The CNN contains a large number of trainable parameters to model the relationship between input and output modalities. Proposed CNN is tested on the Alzheimer's Disease neuroimaging Initiative (ADNI) database. Test results of the CNN has outperformed success with respect to prior methods [5].

In [6], earlier detection of Alzheimer is aimed by analyzing MRI. MRI and functional MRI (fMRI) are the most common diagnostic tools for Alzheimer's disease in clinical research. Detection of Alzheimer's disease is done by considering the similarity of Alzheimer's disease MRI data and standard healthy MRI data. This task can be done by a deep CNN. Brain tumor patients with glioma requires essential treatment planning with accurate prognosis. Conventional survival prediction generally utilizes clinical information and limited handcrafted features from MRI. This is a time consuming and subjective diagnostic approach. In [7], a deep learning framework to automatically extract features from multi-modal preoperative brain images of high-grade glioma patients is developed. Experimental results demonstrate that the approach can achieve an accuracy as high as 89.9%. This result shows the importance of CNN for functional neuro-oncological applications. In [8], CNN is applied to detect multiple sclerosis (MS) pathology. Although changes in brain morphology and white matter lesions are two hallmarks of MS, their variability beyond volumetric is poorly characterized. In this work, CNN classification results show very high accuracy on automatically discover the classic patterns of MS pathology. In [9], Management of Parkinson's Disease (PD) utilizing deep learning is aimed. Although automatic assessment in PD has been studied, so far, no reliable approach has been devised for clinical practice. In the work, an assessment system that abides practical usability constraints and applies deep learning to differentiate disease state in data collected in naturalistic settings is proposed. Results show that deep learning outperforms other approaches in generalization performance, despite the unreliable labelling characteristic for the problem setting.

Mammographic risk scoring has commonly been automated by extracting a set of handcrafted features from mammograms and relating the responses directly or indirectly to breast cancer risk. A CNN considering breast density segmentation and scoring of mammographic texture inputs is proposed in [10]. Microcalcification is an effective indicator of early breast cancer. To improve the diagnostic accuracy of microcalcifications, a deep learning-based discrimination classifier is proposed in [11]. Proposed method improved the microcalcification classification accuracy to 87.3%. Deep learning solutions to Mammogram analysis are done in [12–14]. All these deep learning implementations achieved effective accuracy performance results.

Diabetes is a chronic condition affecting millions of people worldwide. One of its major complications is diabetic retinopathy (DR). This is the most common cause of legal blindness in the developed world. Early screening and treatment of DR prevents vision deterioration; however, the recommendation of yearly screening is required. In [15, 16], possibility of fully automated deep learning system solutions is proposed for prevention of blindness.

Deep learning solutions of cervical diagnosis [17], body part recognition [18], lung cancer detection [19], neuroimaging [20] and cardiac diagnosis [21] are implemented successfully. In addition to these mentioned applications there exist hundreds of biomedical applications and application possibilities of deep learning systems.

In the following parts of the chapter, first deep learning methodologies utilized in these studies are introduced and are analyzed in detail.

2 Deep Learning Methods in Healthcare

In this section, deep learning methods frequently used in healthcare systems are given. After introduction of the convolution neural networks, autoencoders, deep polynomial network, and hybrid methods are investigated. After then convolutional neural network based methods and techniques used in deep learning are taken into consideration. The introduced methods are chosen in accordance with the studies analyzed in the next chapter.

2.1 Convolutional Neural Networks

CNN is a well-known deep learning structure. A general CNN structure consists of convolutional layers, pooling layers, and fully connected layers. Convolutional layers are used to extract features from high sensory input data while pooling layers are used to reduce the size of the data and fully connected layers serve as a classifier [6, 22, 23].

2.1.1 Convolutional Layers

Convolutional layers are the parts of CNN where convolution operation is executed. This operation is applied by using number of certain sized filters. Each filter is shifted over the image, the activation function is applied to the output and a feature map is created with these calculated values. The effectiveness of the convolution process depends on two concepts. These are receptive field and weight-sharing mechanism. The receptive field of a neuron represents the previous layer neurons to which the neuron is linked. The field size is equal to number of the filters used in the convolution. The receptive field concept significantly reduced the number of parameters used in the artificial neural network. The weight sharing mechanism is that the filters have fixed values during one epoch of shifting operation over the entire image. This provides searching the same pattern in the whole image.

2.1.2 Convolution Operation

The convolution process is performed with constant sized filters. These filters are slid over the image, and the activation function is applied. A feature map is created for each filter with these calculated values. Convolutional filters are trainable parameters. During back propagation, the weights of the filter are trained according to the amount of error that falls on the relevant feature map. Let v_{ij}^{xy} be the value at position (x, y) on the jth feature map in ith layer. v_{ij}^{xy} can be calculated as in (1).

$$v_{ij}^{xy} = \sigma(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}) \tag{1}$$

where σ is the activation function, b_{ij} is the bias value, P_i and Q_i are the sizes of the filter, m is index of the feature map and i is the index of convolutional layer.

2.1.3 Pooling Operation

Pooling operation is used for down-sampling the feature maps. The aim of pooling layer is to reduce the size of the feature maps without losing the required information for classification. Pooling operation also ensures that the extracted features are independent of the pattern scale or orientation. The most popular pooling operations are max-pooling and average pooling operations.

2.1.4 Fully Connected Network

The fully connected network consists of multiple layers including neurons connected to each other. It is generally used as a classifier in CNN structures.

2.2 Autoencoders

Autoencoder is the name of the network structure developed for data encoding. Deep autoencoders consists of convolution and pooling layers. While features are extracted with convolutional part with these layers, obtained features are decoded by using upsampling or deconvolutional layers. Figure 1 shows an autoencoder structure used for reconstruction.

There exist different types of autoencoders, some of them are described in the following subsections.

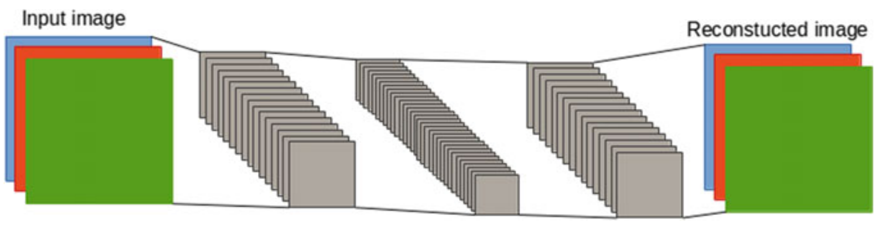


Fig. 1 Use of autoencoder for reconstruction

2.2.1 Sparse Autoencoder

Sparsely connected autoencoder structure is similar to fully connected autoencoder, however, connections between the layers are sparse.

2.2.2 Stacked Autoencoders (SAEs)

SAE is name of a structure composed of two or more autoencoders whose outputs are connected sequentially [24].

2.2.3 Denoising Autoencoders

Unlike the traditional autoencoders, denoising autoencoders aim to generate a good representation even for corrupted input data. Training algorithm is modified to accomplish this goal. Instead of using the original images in the training, randomly distorted images are used in the encoding process as inputs. This structure is then trained to produce the original data from this encoded data obtained from noisy input. Let's x be the original image, a distorted version, $\tilde{x}$, is generated. This generated image is passed through the autoencoder. In the decoding process, the error between the original image and decoded image is used to train the weight values [25]. Figure 2 shows structure of denoising autoencoder.

2.2.4 Stacked Denoising Autoencoders (SDAE)

SDAEs are denoted structures consisting of two or more denoising autoencoders whose outputs of each autoencoder are connected to next one.

2.3 Deep Polynomial Network

Deep Polynomial Network is a neural network structure in which each hidden layer is responsible for a different level of polynomial learning. A 4-level polynomial network is given in Fig. 3.

2.3.1 Stacked Deep Polynomial Network

Stacked Deep Polynomial Network is composed of two or more Deep Polynomial Networks connected sequentially.

2.4 Hybrid Methods

2.4.1 Convolutional Autoencoder Neural Network

Convolutional Autoencoder Neural Network (CANN) structure utilizes unsupervised learning on autoencoder to optimize the parameters of the convolutional layers. This structure requires a small amount of labelled data for fine-tuning operation.

Convolutional autoencoders are used for reconstructing input images by using generated feature maps after the convolution operation. This process is called as inverse convolutional operation. Let $f(.)$ be the convolutional encode operation and $f'(.)$ be the convolutional decode operation, n is the number of $l \times l$ sized feature maps, convolutional encoding operation is executed as follows:

$$o_{ij} = f(x_i) = \sigma\left(w_j.x_i + b\right) \tag{2}$$

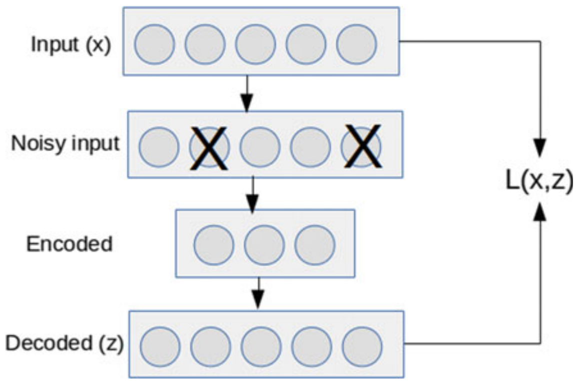
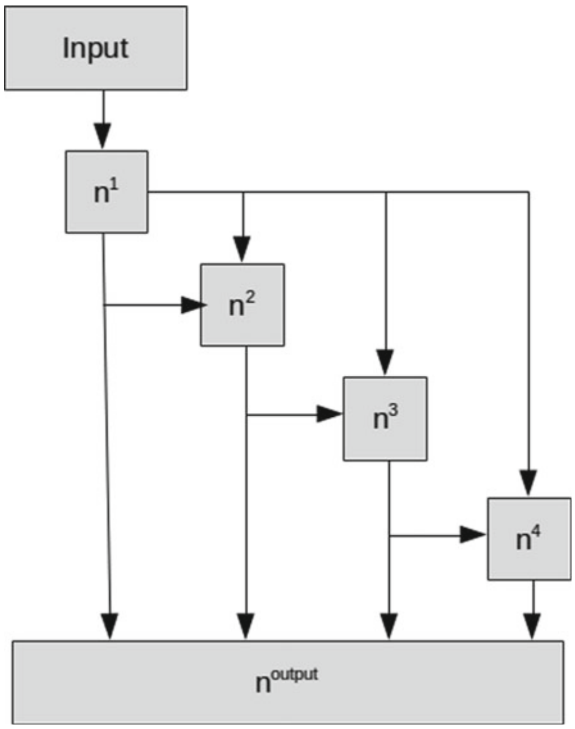


Fig. 2 Structure of denoising autoencoder

Fig. 3 4-level polynomial network



where w denotes the weight, b is the bias value of the corresponding filter, x is the input image, and σ is a nonlinear activation function such as ReLu, sigmoid or hyperbolic tangent. Decoding operation is carried out as follows:

$$\hat{x}_i = f^{'}\left(o_{ij}\right) = \emptyset\left(\hat{w}_j . o_{ij} + \hat{b}\right) \tag{3}$$

After generating a certain number of patches, the corresponding parameters are trained using the squared error between the original images and the reconstructed images with SGD algorithm [26].

2.4.2 Convolutional Recursive Neural Network

Convolutional recursive neural network (CRNN) is another unsupervised CNN structure. CRNN training consists of three steps: pretraining the convolutional neural network, generating local representations, and learning hierarchical feature representations.

- Pretraining CNN filters: Number of different unsupervised methods such as sparse autoencoding, sparse restricted Boltzmann machines, k-means clustering, gaussian mixtures can be utilized to learn convolutional filters. The best performing method in the literature has been shown as k-means clustering. In this operation, number of image patches are randomly selected. After that k-means clustering algorithm is used for minimizing sum of squared Euclidean distances between these patches. In this way, k filters $\{f_k, k = 1, 2, \ldots k\}$ are learned as follows:

$$f_k(x) = \begin{cases} 1, & 1, if\ k = argmin_j ||m_j - x|| \\ 0, & otherwise \end{cases} \tag{4}$$

- Generating local representations: This part of the algorithm simply passes the images through the CNN structure to generate the representations.
- Learning feature representations: Recurrent neural network (RNN) is utilized in this part of the network. RNN provides learning hierarchical feature representations by implementing recursion in a neural network structure. Backpropagation algorithm is used for training the RNN structure [27].

2.5 Convolutional Neural Network Based Methods

AggNet, Spatially Constrained CNN, Temporal Fusion CNN, Triplanar CNN, Regions with CNN, Class Structure based DCNN, Patch wise CNN, Semantic wise CNN, Cascaded CNN, Hough Voting with CNN, U-Net are given in this part. Techniques used in deep learning methods are also given. They are dropout, data augmentation, and finetuning. The most popular pretrained deep learning models LeNet, AlexNet, GoogLeNet, VGGNet, ResNet, and DenseNet are briefly introduced through the subsections.

2.5.1 AggNET

AggNET is a network developed for crowdsourcing applications. The difference between this structure and the traditional CNN structure is that it contains a crowdsourcing layer after the output layer. This layer determines the appropriate output values for the network structure by applying the Expectation Maximization algorithm to the votes received from the users about the unlabeled data. Using data augmentation has also been shown as a way to improve the effectiveness of this structure [28].

2.5.2 Spatially Constrained Convolutional Neural Network

Spatially Constrained CNN (SC-CNN) is a network developed to detect cancerous structures. In order to detect the nuclei of the cancerous structure, the output layer of this structure is designed in two dimensions. While the traditional CNN ends

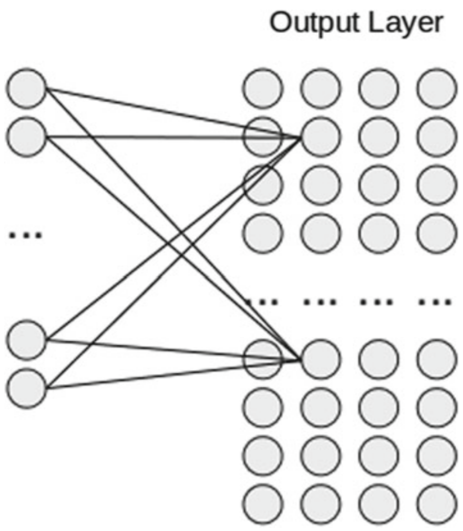


Fig. 4 Architecture of SC-CNN

with a softmax layer, in this structure, outputs are handled in 2 dimensions and each output neuron represents the possibility of being a cancerous cell nucleus. Probability maps are used based on the proximity of the real nucleus during training. Figure 4 represents architecture of SC-CNN.

2.5.3 Temporal Fusion CNN

In contrast to fully visible images on a single layer, data, such as EMR, varies over temporal extent. Temporal connectivity is an important factor for prediction. For this reason, training data is created as bag of samples containing frames for a short fixed-sized time interval. This provides learning of temporal features. Temporal Fusion CNN models are based on the fusing information across temporal domain either by modifying the first convolutional layer to extend in time or by placing two separate single-frame networks and fusing their outputs after in the processing. Figure 5 shows structure of the temporal fusion CNN.

2.5.4 3D CNN

The traditional CNN structure is designed to work on 2D images. In contrast, 3D CNN structure works on 3-dimensional data. Convolutional filters are 3-dimensional and apply 3-dimensional convolution operation. Let v_{ij}^{xyz} be the value at position (x, y, z) on the jth feature map in ith layer. 3D convolution process can be defined as:

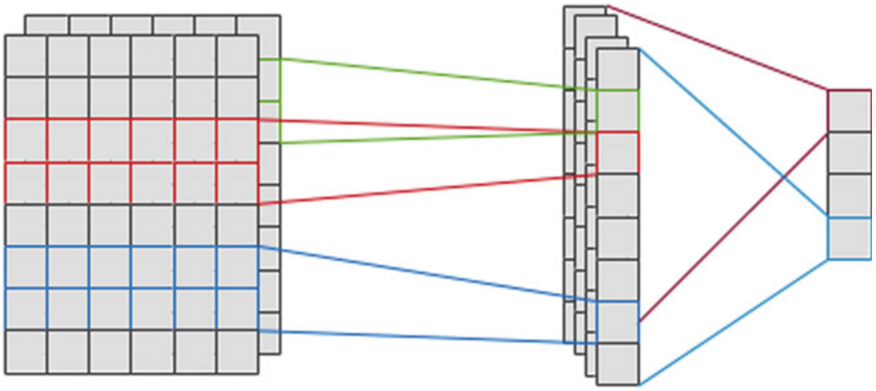


Fig. 5 Temporal fusion CNN

$$v_{ij}^{xyz} = \sigma(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}) \tag{5}$$

where σ is the activation function, b_{ij} is the bias value, P_i, Q_i, and R_i are the sizes of the 3D filter, m is index of the feature map and i is the index of convolutional layer [29].

2.5.5 Triplanar CNN

Using 3D CNN for 3-dimensional data greatly increases the number of parameters of the neural network structure. Triplanar CNN structure handles 3D data by turning it into 2D for each dimension. Three 2D-CNN structures are designed for each dimension (xy, xz and yz spaces), and 2D planes in each dimension are passed through these networks as a patch. These three networks are connected only in output layers [30]. Figure 6 denotes Triplanar CNN architecture.

2.5.6 Regions with CNN (R-CNN)

R-CNN is a CNN structure utilizing SVM approach rather than fully connected network for classification.

2.5.7 Class Structure Based DCNN (CSDCNN)

CSDCNN is a deep neural network structure providing non-linear representations. This method combines the feature extraction process with feature learning. Therefore, it does not require hand-design features. CSDCNN is able to learn discriminative and semantic features for all levels [31].

2.5.8 Patch Wise CNN

Patch-Wise CNN is a simple method used for segmentation. An NxN patch around each pixel inside the image is taken and CNN model is trained with these patches.

2.5.9 Semantic Wise CNN

Semantic-Wise CNN contains convolutional, pooling, upsampling and deconvolution layers similar to autoencoders and makes predictions for each pixel inside the image for segmentation [32].

2.5.10 Cascaded CNN

Cascaded CNN is composed of two CNN structures. The output of the first CNN is the input for the second one.

2.5.11 Hough Voting with CNN

Hough voting with CNN is an efficient segmentation method based on a voting strategy. It is a multi-modal, multi-region and robust. It considers not only categorical predictions extracted by CNN but also features extracted in intermediate layers. These features are useful for tasks such as image retrieval [33].

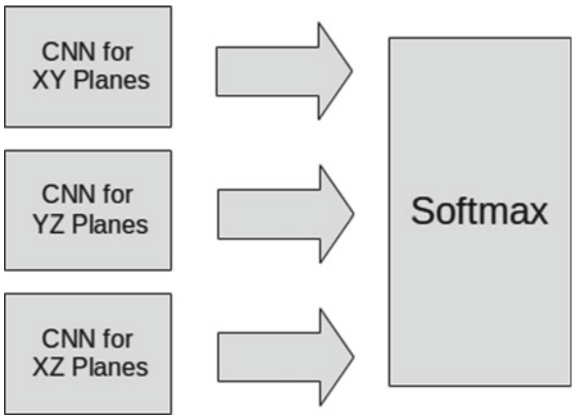


Fig. 6 Triplanar CNN Architecture

2.5.12 U-Net

U-net is a novel architecture designed for segmentation. “U-net” stands for “U” shape of the structure. First part of the network consists of convolution and pooling layers while the second part contains upsampling operations [34].

2.6 Techniques Used in Deep Learning Methods

Overfitting is the most important problem for the machine learning algorithms. Moreover, deep learning algorithms require more data and more training epochs to provide efficient results. There are some techniques to solve these problems. In this section, these techniques are discussed.

2.6.1 Dropout

Dropout is a technique developed to avoid overfitting in neural networks. In this method, activations of some randomly chosen neurons are taken zero during the training. For each iteration, different neurons are selected randomly. This allows a more reliable training process.

2.6.2 Data Augmentation

Data augmentation is artificially differentiating the data. Generated artificial data is also included in the training set. This provides a better generalization by allowing objects to be recognized independently of their size and position.

2.6.3 Fine Tuning

Finetuning operation is the training of a previously trained deep learning network for a new set of data. This operation significantly shortens training time.

Popular Pre-trained Deep Learning Models

In this section, the pre-trained deep learning models used as finetuning dataset for the biomedical applications are explained.

LeNet

LeNet was proposed by LeCun in 1998. Although it is simple according to the current deep learning models, it is important to be the first of its kind. In this structure, two convolution and pooling layer pairs are followed by a fully connected network structure. This structure was used for the classification of MNIST dataset [4]. Its structure is given in Fig. 7.

AlexNet

AlexNet is a well-known deep learning structure. It showed the best performance in ILSVRC2012 competition by classifying the ImageNet with 16.4 top5 error rate. ImageNet is a very popular vision dataset containing 1.3 million images divided into 1000 classes. AlexNet has a similar structure with LeNet, however it has two additional parallel CNN structures containing more filters. These parallel CNNs are end up with a fully connected classifier [30, 35]. Figure 8 shows the Alexnet structure.

GoogLeNet

GoogLeNet is the winner of ILSVRC2014 competition. Top-5 error rate of GoogLeNet was 6.67% on ImageNet dataset. While GoogLeNet is similar to LeNet, it includes a novel module called Inception. This module contains convolution operations with different sized kernels and pooling operations. GoogLeNet with 22 layers has 4 million parameters and has performed much better than AlexNet containing 60 million parameters [30].

VggNet

VggNet is another popular model participated to ILSVRC2014. It contains 16 convolutional layers applying 3 × 3 convolution operations. The biggest problem of VggNet is that it includes 138 million parameters to be trained [30]. Figure 9 represents the structure of VggNet.

ResNet

ResNet (Resudial Neural Network) is the winner model at the ILSVRC2015 competition. It achieved a top-5 error rate of 3.57% on ImageNet dataset which is better than human-level performance. What makes ResNet different from other neural network structures is its "skip connections" concept. A skip connection denotes merging the input of the convolutional block with its output by adding them together.

DenseNet

DenseNet was developed with a similar idea to ResNet. However, its connections do not only link the previous layer to the next layer but also all subsequent layers.

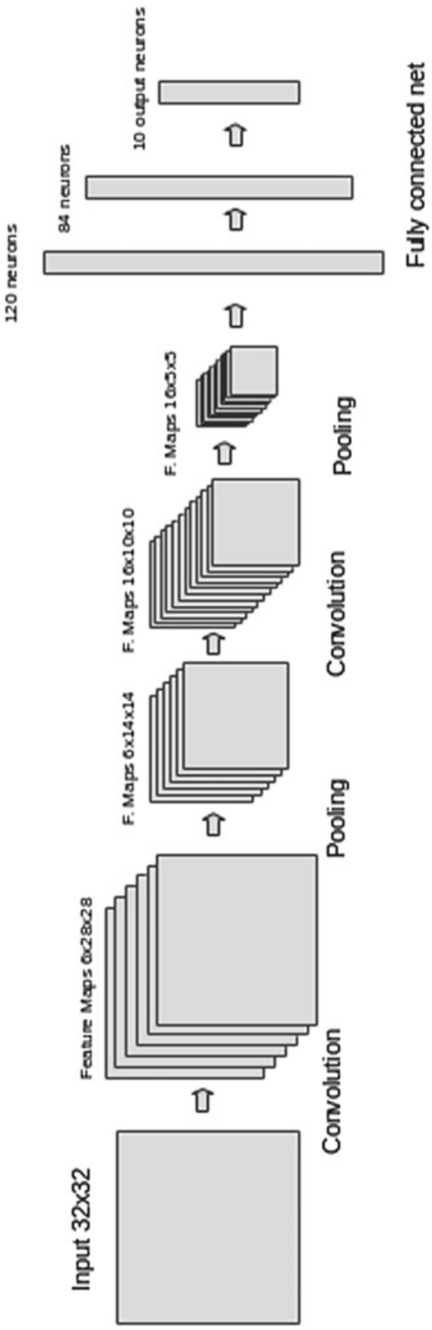


Fig. 7 Architecture of LeNet

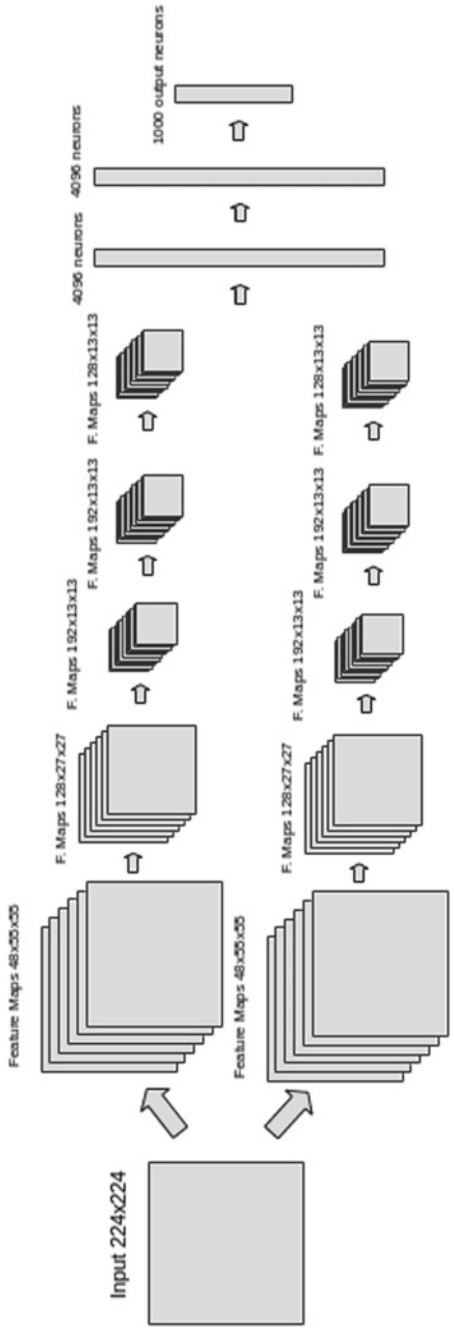


Fig. 8 AlexNet

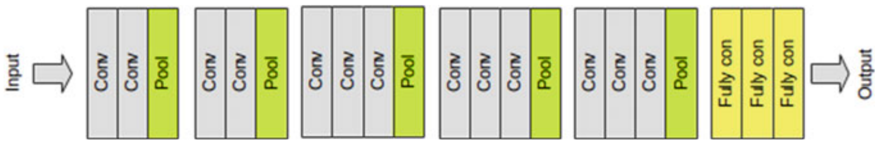


Fig. 9 VggNet

3 Deep Learning Applications in Healthcare

Although biomedical science contains many sub-areas for Deep Learning implementations, the most popular of them are selected and given in the following parts. These are mainly classified as breast and breast cancer related applications, brain imaging and diagnosis related applications, diabetic related applications, cardiac related applications and lung cancer related applications. Through all of these applications Deep Learning systems are adapted to the nature of the medical problem and their estimations are evaluated with known and experienced test data.

3.1 Breast Mass and Microcalcifications Detection for Breast Cancer Classification

Mammography is an efficient imaging technique used for detecting masses and microcalcifications. Masses are gray or white regions in the breast area with oval, irregular, or lobulated shapes. Mass segmentation is an important step for cancer detection [36].

In [37], masses are detected by combination of deep learning and random forests approach. First, candidate mass regions are determined by using cascade of multi-scale deep belief network classifiers with Gaussian mixture model. Then, correct regions are detected by a cascade of deep convolutional neural networks consisting of convolutional layers followed by linear SVM. After obtaining regions, morphological and texture features are extracted from correct regions and classified by cascade of random forests classifiers. While the method detects all mass candidates at the first step, final true positive rates are 0.96 for INbreast dataset, and 0.75 for DSM-BCRP dataset. Authors propose two potential functions: conditional random field (CRF) and structured support vector machine (SSVM) on their previously proposed deep structure, respectively. It is reported that dice indexes obtained with CRF and SSVM are 0.9 [37].

In [13], Bayesian optimization is applied for the mass detection step. After applying multi-scale deep belief networks and Gaussian mixture model classifier, false positives are reduced by cascade of CNN and RF. At the last part, Bayesian optimization is applied to refine the results. While dice index of segmentation performance is 0.85, overall classification performance of masses as malignant or benign is 0.98 [13] on INbreast dataset.

In [38], CNN based mammogram analysis is introduced to detect the risk of breast cancer development. Craniocaudal and mediolateral oblique views, their mass and microcalcification segmentations are applied to separate CNN models pre-trained with ImageNet. Obtained features are used for training a final CNN model to predict patient's risk in terms of Breast Imaging-Reporting and Data System (BI-RADS) score. INbreast and DDSM datasets are used for testing. It is reported that model achieve AUC of 0.91 on INbreast dataset and 0.97 on DDSM for two-class (benign or malignant) problems. Another transfer learning based method is utilized in [35] to detect benign and malignant lesions. Pre-trained CNN features and analytical features obtained from segmented tumors are classified with polynomial kernel SVMs on dataset obtained from University of Chicago Medical Center. Voting mechanism is applied for the final result. While AlexNet is used as pre-trained model, classification abilities of the features obtained from AlexNet's layers are compared. Features are flattened and SVM classifier is applied for each layer of the AlexNet. Features from the fourth convolutional layer provide the highest AUC $= 0.83$ and for the overall performance, ensemble method provides better than the analytical approach with 0.86 AUC [35]. Levy and Jain proposed CNN based technique to classify pre-segmented breast masses in mammograms. Their technique includes transfer learning approach utilizing AlexNet and GoogLeNet, and pre-processing step providing mass context and data augmentation. Three different CNN models are compared on Digital Database for Screening Mammography (DDSM) dataset which is public and provided by University of South Florida. The first CNN model is baseline model similar to the early versions of AlexNet, the second and third models are AlexNet and GoogLeNet. Convolutional weights of second and third models are initialized with pre-trained model weights and fine-tuned with mammography images. Inputs to these networks are obtained by pre-processing technique. Since area around the mass is crucial, two different patches are obtained from the images. In the first patch, inputs are regions with 50 pixels of fixed padding around the mass. In the second patch, inputs are regions two times the size of the mass bounding box. Dataset is augmented by applying transformations such as rotation, cropping, and mirroring. According to the reported results, pre-processed and augmented GoogLeNet provides the highest accuracy 0.929 on the test set while pre-processed and augmented baseline and AlexNet models provide 0.6 and 0.89, respectively [23].

Microcalcification plays important role in early breast cancer detection [11]. SAE model is utilized to measure discrimination power of the microcalcification in breast cancer detection and compared with other state of art machine learning approaches such as LDA, KNN, and SVM. Three different experiments are tested to demonstrate the importance of microcalcification. Dataset includes mammograms from 1204 female patients, 774 of them diagnosed with benign and 430 of them diagnosed with malignant breast lesions, at the Sun Yat-sen University Cancer Center (Guangzhou, China) and Nanhai Affiliated Hospital of Southern Medical University (Foshan, China). In the first experiment, only extracted microcalcification data are used. In the second experiment, only breast masses data are used and in the last experiment, breast masses data combined with microcalcification data are used. Suspicious breast masses and microcalcification data are extracted by using image segmentation. Their statistical and texture features are applied to classifiers. It is reported that the best classification performance is achieved with SAE and while 0.87 AUC is obtained by using only microcalcification data having 15 features, 0.61 AUC is obtained by using only breast mass data having 26 features and combination of microcalcification and breast mass data provides 0.9 AUC [11].

Unlike presented studies above, in [12] large scale machine learning approach is applied to detect mammographic lesions and a convolutional neural network is trained on a dataset including 45,000 images. Before training convolutional neural network, candidate regions are detected with two-stage classification approach. At the first step, first and second order Gaussian kernels are applied to extract five features representing center of a focal mass, speculation patterns, the size of the optimal response in scale-space. At the second step, random forest is applied to all pixels representing estimated suspiciousness. All obtained optima in the likelihood image by using non-maximum suppression are applied as inputs to the CNN. Data augmentation is provided by performing translation and scaling. Trained CNN architecture is similar to OxfordNet structure and comprised layers of 16, 32, 64, 128, 128 with 3×3 kernels with 2×2 max pooling operator. Convolutional layers are followed by two fully connected layers with 300 neurons. Performance of the structure is compared with a reference system based on candidate detector, contrast, texture, geometrical and location features. Moreover, features obtained from CNN combined with each feature used in reference system. According to the given results, while CNN trained on augmented data provides 0.929 AUC, CNN combined with other features provides 0.941 AUC.

Deep learning structures provide better performances on large datasets, however collecting annotated large dataset is a challenging problem in many areas. In [28], crowdsourcing layer is added to convolutional neural network and named as AggNet. The problem with the crowdsourcing is noisy annotations. Hence, in AggNet, data aggregation is made as a part of training phase. Multiple CNNs are trained on different image scales to detect mitosis. Obtained mitosis candidates are sent to crowds for annotations. Annotated images are sent to CNN with aggregation layer to correct the model and generate a ground-truth data. CNN used in this study consists of three convolutional layers and two fully connected layers. Aggregation layer includes majority voting technique. After aggregation, sensitivity and specificity measures are calculated and EM algorithm is utilized for training. Proposed structure is tested on MICCAI-AMIDA13 challenge dataset and compared with augmented models. It is reported that AggNet provides the highest AUC, which is 0.86. Moreover, AggNet is trained on 8 random patients' data and tested on 3 different patients and reported F1 score is 0.7419. Another experiment is on whole AMIDA13 dataset and obtained overall F1 score is recorded as 0.43.

Wang et al. won the competition launched at The International Symposium on Biomedical Imaging (ISBI) to detect metastatic breast cancer. Challenge had both whole slide image classification and tumor localization competitions on Camelyon16 dataset, and their AUC scores are 0.925 and 0.7051, respectively. While pathologist's predictions have 0.966 and 0.733 AUC scores, combination of pathologist predictions and deep learning systems provides 0.995 AUC score for the classification.

Threshold-based segmentation deep learning system consists of preprocessing, patch-based classification stage and heatmap-based classification stage. At the patch-based classification stage, randomly extracted positive and negative regions at different magnification levels are fed to GoogLeNet, AlexNet, VGG16, and FaceNet. According to the reported results best performance 98.4% is obtained from GoogLeNet at $\times 40$ magnification level. On the other hand, tumor probability heatmap obtained from GoogLeNet demonstrates that false positive samples cause the majority of the error. Hence, more hard negative patches are sampled, and model is retrained to eliminate this problem [30].

Generally, in the literature, breast cancer classification systems are designed to separate benign and malignant cases. In [31], class structure-based deep CSDCNN is proposed to identify subordinate classes such as Ductal carcinoma, Fibroadenoma, Lobular carcinoma, etc. CSDCNN includes input layer, convolutional layer, and pooling layer. CSDCNN modifies the loss function by adding a constraint for feature space to preserve intra-class variance. CSDCNN is tested on BreaKHis dataset including 7909 images representing eight sub-classes of breast cancers. Classification performances of AlexNet, LeNet and CSDCNN are compared at image and patient level. According to the reported results, CSDCNN with data augmentation provides 93.9% for image level classification with $100\times$ magnification factor and 94.7% for patient level classification with $200\times$ magnification factor.

In [39], two different methods utilizing U-net are proposed to segment fibroglandular tissue (FGT) MRI volumes. While first method (2C) segments breast in the whole MRI, and FGT inside the segmented breast; second method (3C) segments MRI volume into non-breast, FGT inside breast, and fat inside breast regions. Hence, second method provides two segmentation tasks simultaneously. Presented structures are tested on 66 MRI data with four different breast density levels and compared with image processing techniques such as atlas and sheetness. According to the reported results, best results are obtained for least dense breasts and while 2C provides 0.944 dice similarity coefficient, 3C provides 0.933 dice similarity coefficient.

In [10], an unsupervised method called CSAE is proposed to extract features from unlabeled data for mammographic risk scoring which is calculated using features extracted from mammograms. The method consists of two steps: breast density segmentation and scoring. The main idea behind the method is mapping multichannel image patch with size (c, m, m) to a patch with size (C, M, M) having one channel per label. Unlike direct mapping, in this method L layers are used for mapping inputs to labels, called as feature encoding. Classifier part classifies last feature representation into label space.

Another study utilizing CNN to detect invasive tumors on whole slide images, includes tile tissue sampling, tile-preprocessing, and classification steps. Tile tissue sampling is the process of extracting fixed size regions. Tile-preprocessing includes color space conversion and color normalization. Classifier is trained on tile regions annotated by pathologists. 3, 4, and 6 layered convolutional neural networks are evaluated, and 3-layer convolutional neural network provides the best AUC score, which is 0.9018 [22].

3.2 Brain Tumor Segmentation and Alzheimer's Disease

Brain tumor segmentation is one of the crucial tasks in medical diagnosis systems. Therefore, brain imaging community provided many publicly available datasets and different techniques are utilized to obtain accurate results. MICCAI 2008, Brain Tumor Segmentation (BRATS), Ischemic Stroke Lesion Segmentation (ISLES), Mild Traumatic Brain Injury Outcome Prediction (mTOP), Multiple Sclerosis Segmentation (MSSEG), Neonatal Brain Segmentation (NeoBrainS12), and MR Brain Image Segmentation (MRBrainS) datasets are some of the publicly available datasets. Generally, preprocessing steps such as spatial alignment, skull stripping, contrast correction, intensity normalization, and noise reduction are required to be able to analyze the MR images. Patch-wise CNN, semantic-wise and cascaded CNN are some of the modifications proposed on tradition CNN structures [32, 40]. In [20], 25 papers introducing deep learning methods and applications utilized to analyze the neuroimaging correlates of psychiatric and neurological disorders are reviewed in detail.

In [41], fully convolutional neural network and conditional random fields are combined to segment brain tumors. 2D images and image slices from axial, coronal, and sagittal views are trained with fully convolutional neural network and conditional random fields as recurrent neural networks, respectively. Brain images are segmented slice by slice via voting technique. Two different sized image patches obtained from slices are applied to fully convolutional network to generate feature maps. Larger inputs are transmitted over series of convolutional and pooling layers until they reach the same size with small inputs. Then, both feature maps generated from small and large inputs are applied to remain convolutional and pooling layers. After fully convolutional network process, conditional random fields using the weights of trained fully convolutional network segments the image by minimizing the energy function. At last, both parts are finetuned. Proposed technique is tested on Multimodal Brain Tumor Image Segmentation Challenge (BRATS) 2013, BRATS 2015 and BRATS 2016. According to the results, 0.84 dice coefficient is obtained for BRATS 2015.

In [33], brain regions are segmented by applying Hough voting mechanism on outputs of six different convolutional neural network structures. Both ultrasound and MRI data are utilized to train structures. While MRI dataset has 55 subjects obtained via 3D gradient echo imaging, ultrasound dataset consists of 162 volumes obtained by several freehand 3D scans recorded through windowing on 34 subjects. 80 K and 400 K sized training sets are used for ultrasound and 114 ultrasound volumes are used for testing. For MRI data, 10 volumes are used for testing. CNN is trained on 8 subjects, tested on 24 subjects and validated on 2 subjects. 2, 2.5, and 3-dimensional data are trained separately. Hough-CNN method provides the best result (0.85 dice coefficient) on ultrasound dataset. 7-5-3 CNN structure on 3D data provides the best result (0.77 dice coefficient) on MRI data.

In [29], deep CNN structure is utilized to obtain 3D feature representations of MRI. One of the important properties of the study is that rather than using 2D convolutional filters directly for 3D medical imaging modalities, high dimensional features are concatenated as 3D features to eliminate overfitting problem. Study consists of three steps: cerebral microbleed candidate localization accomplished by analyzing the intensity distribution and thresholding, 3D feature representation with CNN, classification with SVM. From 20 subjects 117 cerebral microbleeds are used for training and testing. Reported test results demonstrates that while CNN and RF based solutions provide 0.5405 and 0.5031 F1 measures respectively, 3D feature-CNN provides 0.6891 F1 measure [29].

Multiple sclerosis (MS) is a complex pathology to be analyzed. Changes in brain morphology and white matter lesions are important symptoms [8]. In [8], deep belief network based model is proposed to detect changes in brain morphology and lesion distribution. Model consists of three elements detecting morphological changes, discovering spatial distribution of lesions, and joint model. The model is tested on a dataset obtained from an MS clinical trial of 474 secondary progressive MS patients and containing T1w, T2w, and PDw MRI data for each patient. Before processing, rigid registration, brain extraction, intensity normalization and background cropping are applied. Pearson correlation of the clinical MS Functional Composite and its elements with the distribution parameters and two established MS imaging biomarkers are calculated to evaluate potential of the distribution parameters to find relevant information.

Multi-modality may provide better diagnosis systems, however the main issue in the multi-modality is incomplete data. In [5], convolutional neural network based deep learning model for estimating the multi-modality imaging data is proposed. The idea behind the model is to reveal the relationship between input modality and output modality. First, intensity inhomogeneity correction, skull-stripping, and cerebellum removing steps are applied on T1w MRI. MR images are segmented into cerebrospinal fluid, white matter, and gray matter, and spatially normalized. PET images are also aligned to the related MR images. Gaussian kernel is used to smooth gray matter tissue density maps and PET images. Patches extracted from MRI and PET are applied to 3D CNN architectures. There are two 3D CNN layers including 10 filters with $7 \times 7 \times 7$ size and one feature map at the output layer. Model is tested on ADNI database and while MRI images are inputs, PET images are outputs of the model. 830 subjects from ADNI database are utilized for the evaluation and 432 of them does not have the PET images. 3D CNN model is compared with KNN and Zero methods. According to the reported results, 3D CNN model outperformed KNN and Zero methods on both training and test samples.

Alzheimer's disease (AD) is a disease that decreases quality of life and is frequently seen in society. Early diagnosis of AD is important to eliminate the brain damage and depends on identifying risk of Mild Cognitive Impairment (MCI) progression. In [24], 3 layered stacked autoencoders and softmax layer are utilized to classify AD. Grey matter volumes and CMRGIc patterns are extracted from MRI and PET, respectively. Features are selected by using Elastic Net and normalized to zero mean and between 0 and 1. Method is evaluated on ADNI dataset including MRI of 311 subjects. According to the reported test results, for AD versus normal control (NC) classification problem, the method provides 87.76% accuracy while single kernel SVM (SK-SVM) and multi kernel SVM (MK-SVM) provides 84.40% and 86.42%, respectively. For MCI versus NC, the method provides 76.92% accuracy, SK-SVM and MK-SVM provides 76.81% and 77.25%, respectively. The method provides the best accuracy (47.42%) for 4-class problem. In [42], deep belief network based classification approach is proposed to obtain hierarchical feature representations from 3D patches and design joint features from paired patches of MRI and PET. Anterior Commissure (AC)–Posterior Commissure (PC) correction, skull-stripping, and cerebellum removal processes are applied to MRI data and then MRI data is segmented to gray matter, white matter, and cerebrospinal fluid and spatially normalized to be able to align with MNI coordinate space. Regional volumetric maps are generated. Gaussian kernel is utilized to smooth gray matter density maps and PET images. First, class discriminative patches are extracted from MRI and PET images, and selected by using statistical significance test between classes. Patch level feature learning is provided by stacked restricted Boltzmann machine (RBM). For each obtained patch, a linear SVM classifier is built and its outputs are converted to a probability via softmax function. Locally distributed patches are obtained by constructing spatially distributed mega-patches generated in greedy manner. Mega-patches are classified by classifiers constructed in greedy manner. At last, multiple image level classifiers are built by fusing the mega-patch classifiers. Fusing process is provided by selecting optimal subset of mega-patches in greedy search strategy. Multiple classifiers select possibly a different subset of mega-patches. Selected frequency of mega-patches in each image-level classifier is counted, normalized and used as weights of the respective mega-patches. Weighted combination of outputs of the mega-patch classifiers provides the final decision. Approach is evaluated on ADNI database baseline MRI and (FDG-PET) data obtained from 602 subjects. According to the reported results, AD versus NC classification accuracy is maximally 95.35%, MCI versus NC classification accuracy is 85.67%, and MCI converter versus MCI nonconverter classification accuracy is 74.58% [42].

In [43], effectiveness of the different convolutional architectures such as stacked 2D patches, 2D patches, tri-planar, and 3D convolution are compared to segment 3D hippocampal patches for AD diagnosis. ADNI dataset is utilized for evaluations. Best performances are obtained by using 3D convolution and tri-planar approach.

Another approach adopting CNN architecture, LeNet5, to distinguish healthy brain and Alzheimer's brain is trained and tested on fMRI data of ADNI database. Before the classification, nonbrain tissues are removed from T1 images, motion correction (MCFLRIT), skull stripping, and Gaussian kernel spatial smoothing are applied as pre-processing steps. Obtained average classification performance is 96.8588% [4].

In [6], ensemble CNNs are proposed to classify healthy MRI and AD MRI. Three CNN architectures including several convolution, pooling, transition layers and dense block are utilized. Transition layer consists of batch normalization, 1×1 convolution, and 2×2 pooling layers. Dense block denotes combination of dense layers representing connectivity pattern between layers. These layers have small sized filters and each layer is connected to the other. This provides global feature map set. Since the proposed model has 2D structure, 3D MRI data is converted to 2D images by creating patches from axial, coronal, and sagittal planes. Obtained patches are used as inputs. Output layer has four classes denoting non-demented, very mild, mild, and moderate AD. Five different models and ensembled model are created and compared with ResNet and Inception-v4. According to the reported results, ensembled model provides 0.902 f1-score while Inception-v4 and ResNet provide 0.71 and 0.75 f1-scores, respectively.

3.3 Diabetic Retinopathy and Cataract

Cataract and diabetic retinopathy are two leading causes of blindness. Early diagnosis of them may eliminate vision degradation [15, 27]. Although automatic methods exist in the literature, they have problems such as incomplete and redundant images, noisy representation. In [27], severity of nuclear cataracts is automatically graded by applying deep learning methods on slit-lamp images. Method consists of structure detection, feature learning, and classification steps. In structure detection step, lens is localized by detecting visual axis of the lens through edge detection and circle fitting algorithms. Then, it is divided into three regions: anterior cortex, nucleus, and posterior cortex. However, anterior cortex is removed from the set. Next step is resizing nucleus and posterior cortex regions. Resized nucleus region is also divided into three half-overlapping regions and they are fed to the feature learning step. In feature learning step, unsupervised convolutional recursive neural network is utilized. CNN filters are pretrained and fed into a CNN layer to obtain local representations of each image, and multiple recurrent neural networks are utilized to learn hierarchical feature representations. First, randomly patches are generated, and k-means clustering is applied to obtain a group of local filters. For each section in a category of images one group of local filters is extracted. The last group of filters is learned from all patches over all categories for the given category. Grading is achieved by applying support vector regression on the filters obtained at the feature learning step. Evaluation is done on ACHIKO-NC dataset including 5387 images with grading scores from 1 to 5. According to the given test results, the method provides significant performance increase and mean absolute error is 0.304. In [44], diabetic retinopathy severity is graded by using deep convolutional neural network. Inception-v3 structure is trained on 128,175 macula-centered retinal fundus images. Multiple binary classifications (moderate or worse diabetic retinopathy; severe or worse diabetic retinopathy; referable diabetic macular edema; fully gradable) are performed on single network structure. According to the reported results 97.5% sensitivity is obtained.

SqueezeNet based deep structure is trained for 2015 Diabetic Retinopathy Detection Kaggle Competition. Dataset is provided by EyePACS including 35,126 training retinal fundus images and 53,576 test images. Images are graded using severity levels from 0 to 4. Obtained overall accuracy on test set is 83.6% and Quadratic Weighted Kappa score is 0.76543 [15]. In [45], Messidor-2 dataset is utilized including 1748 images to improve Iowa detection program with CNN detectors. Device provides four output types; negative, referable DR, vision-threatening, low exam quality. According to the reported results, device could predict referable DR with 96.8% sensitivity, vision-threatening with 100%.

Retinal vessel segmentation is an important process for diagnosis systems. Hence, automatic vessel segmentation has become important topic and many methods have been proposed. DeepVessel is one of these studies utilizing convolutional neural network with side output layer and conditional random field methods. Side output layer provides better hierarchical representations while conditional random field models the interaction between pixels. DeepVessel is built on holistically-nested edge detection (HED) architecture and fine-tuned with ARIA and DRIVE datasets. Method is evaluated on publicly available DRIVE, STARE and CHASE_DB1 datasets and 0.9523, 0.9585, and 0.9489 accuracies are obtained, respectively [46]. In [47], deep network structure is trained on up to 400,000 sample of examples. Six different convolution architectures plain, with global contrast normalization, with zero-phase whitening, with augmented, no-pool, and balanced are tested on DRIVE and STARE datasets. The reported results show that balanced architecture provided the best AUC results.

A CNN structure is trained on optical coherence tomography (OCT) images to detect intraretinal fluid. Proposed structure is the modified version of U-net producing a binary segmentation map. 1289 OCT images are divided into train and cross validation sets including 934 and 35 images, respectively. Maximum Dice coefficient obtained from cross validation is reported as 0.911 [34].

3.4 Cardiology

Heart health assessment requires automatic localization and tracking of the left ventricle (LV). Deep learning based approaches have been introduced in the literature to localize LV [48]. In [48], CNN is utilized to localize LV in cardiac MRI. Publicly available dataset including 33 patients' MRI provided by York University is used for training and testing steps. During training step, 19 of them are used and each slice is divided into positive and negative patches. While the positive patch is represented by the smallest bounding box enclosing the LV, 8 neighboring patches of positive patch are used as negative patches and one more negative patch is selected randomly. Totally, 30,000 patches are used for CNN training. Another issue for the localization is variability in the heart size among the patients. Pyramid of Scales Localization (PoS) is utilized during testing phase. Each input image is resized four different scales. At each scale, pixel by pixel scanning is performed and probability map representing the likelihood that a pixel in the image is the center of LV at that scale. Probability maps are resized, and different selection policies based on the pixel having the highest probability are applied to determine the scale. According to the reported results while single scale provides 96.5% accuracy, highest probability over PoS, PoS average probability based selection policy, and PoS average probability and standard deviation based selection policy provides 98.65%, 98.64%, and 98.66% accuracies, respectively.

In [25], electrocardiogram (ECG) signals are classified by using stacked denoising autoencoders (DAE) and softmax layer to label ECG beats. Weights of the structure are updated by allowing expert interaction. Method consists of three steps: unsupervised DAE training, fine tuning of DAE and softmax layer, and fine tuning by expert. MIT-BIH arrhythmia database including 48 two-lead recordings, INCART including 75 recordings, and SVDB including 78 two-lead recordings are used in experiments. These datasets have four classes named as N, S, V, F and experiments are employed as two binary classification problems (V vs. N, S, F (VEB) and S vs. N, V, F (SVEB)). On MIT-BIH, accuracy results are 100 for SVEB and 99.8 for VEB. On INCART database, 99.91 and 99.83 accuracies are obtained. On SVDB, 98.77 and 99.58 accuracies are obtained.

3.5 Lung Cancer Detection

Sensors have become frequently used devices in daily life with the improvements on mobile healthcare systems. In [19], stacked autoencoder method is used for feature extraction from Gas Chromatography Mass Spectrometer (GC-MS) data which represents human urine level to detect lung cancer. The system is trained on 57 patient's urine data and tested on 10 patient's urine data. The highest accuracy obtained from the system is 90%. Autoencoder is utilized as feature extractor in other application classifying the lung nodules as malignant or benign. 4303 samples are used from the National Cancer Institute (NCI) Lung Image Database Consortium (LIDC) dataset. Binary decision tree is used for classification process and 75.01% accuracy is obtained over 10 fold cross validation [49]. Data labelling is a challenging task especially for medical images. In [26], convolutional autoencoder (CANN) structure requiring small amount of labeled data is utilized for unlabeled feature learning. The dataset consists of two types of data: labelled and unlabeled. While unlabeled data is used at feature learning step, labeled data is utilized for fine-tuning and classifier training. The raw CT images are divided into patches and used as inputs to CANN. There exists three convolutional, three pooling layers and one fully connected layer in the structure. CANN includes reconstruction step for the convolutional layer, convolutional conversion from feature map to output is named as convolutional decoder and reconstruction is obtained by inverse convolutional operation named as convolutional encoder. Training is accomplished through standard autoencoder training approach. Proposed structure is trained and evaluated on 4500 patients' lung CT and 95% accuracy is obtained.

4 Conclusion and Future Works

Healthcare systems produce crucial and big data. Hence, techniques utilized to analyze these data should be accurate and efficient. Deep learning is the most applied data analysis method with high representation and feature learning abilities. In this chapter, the most popular and up-to-date deep learning solutions to biomedical problems are mentioned and analyzed according to problem characteristic, importance of solution, requirements, and achievements. First, deep learning methods utilized in healthcare systems are described. It is realized that convolutional neural network is the most promising approach utilized in healthcare systems due to its feature extraction ability. Then, convolutional neural network based approaches are described. Generally, the important challenge encountered in healthcare systems using deep learning is lack of labelled data. Hence, autoencoders are frequently utilized to train parameters of convolutional neural networks due to its unsupervised learning. After describing structures, techniques used for increasing the efficiency of deep learning approaches such as dropout, data augmentation, and finetuning are discussed briefly. At the last part of that section, frequently used pre-trained models are described. Third section of the chapter is devoted to the deep learning applications in healthcare systems and their achievements. According to the reviewed studies, deep learning techniques are efficient, and accurate enough to be used in healthcare systems.

References

1. Miotto, R., Wang, F., Wang, S., Jiang, X., Dudley, J.T.: Deep learning for healthcare: review, opportunities and challenges. Brief. Bioinf. **19**(6), 1236–1246
2. Shen, D., Wu, G., Suk, H.I.: Deep learning in medical image analysis. Annu. Rev. Biomed. Eng. **19**, 221–248 (2017)
3. Brosch, T., Tam, R.: Manifold learning of brain MRIs by deep learning. Med. Image Comput. Comput. Assist. Interv. **16**(2), 633–640 (2013)
4. Sarraf, S., Tofighi, G.: Deep learning-based pipeline to recognize Alzheimer's disease using fMRI data. In: FTC 2016—Future Technologies Conference 2016, 6–7 December 2016, San Francisco, United States, pp. 816–820 (2016)
5. Li, R., Zhang, W., Suk, H., Wang, L., Li, J., Shen, D., Ji, S.: Deep learning based imaging data completion for improved brain disease diagnosis. Med. Image Comput. Comput. Assist. Interv. **17**(3), 305–312 (2014)
6. Islam, J., Zhang, Y.: Brain MRI analysis for Alzheimer's disease diagnosis using an ensemble system of deep convolutional neural networks. Brain Inf. **5**, 2 (2018)
7. Nie, D., Zhang, H., Adeli, E., Liu, L., Shen, D.: 3D deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients. Med. Image Comput. Comput. Assist. Interv. **9901**, 212–220 (2016)
8. Brosch, T., Yoo, Y., Li, D.K.B., Traboulsee, A., Tam, R.: Modeling the variability in brain morphology and lesion distribution in multiple sclerosis by deep learning. Med. Image Comput. Comput. Assist. Interv. **17**(2), 462–469 (2014)
9. Hammerla, N.Y., et al.: PD disease state assessment in naturalistic environments using deep learning. In: AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, pp. 1742–1748 (2015)

10. Kallenberg, M., et al.: Unsupervised deep learning applied to breast density segmentation and mammographic risk scoring. IEEE Trans. Med. Imaging **35**(5), 1322–1331 (2016)
11. Wang, J., Yang, X., Cai, H., Tan, W., Jin, C., Li, L.: Discrimination of breast cancer with microcalcifications on mammography by deep learning. Sci. Rep. **6**, 27327 (2016)
12. Kooi, T., Litjens, G., Ginneken, B., Gubern-Merida, A., Sanchez, I.C., Mann, R., Heeten, A., Karssemeijer, N.: Large scale deep learning for computer aided detection of mammographic lesions. Med. Image Anal. **35**, 303–312 (2017)
13. Dhungel, N., Carneiro, G., Bradley, A.P.: A deep learning approach for the analysis of masses in mammograms with minimal user intervention. Med. Image Anal. **37**, 114–128 (2017)
14. Litjens, G., et al.: Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. Sci. Rep. **6**, 26286 (2016)
15. Lunscher, N., Chen, M.L., Jiang, N., Zelek, J.: Automated screening for diabetic retinopathy using compact deep networks. J. Comput. Vision Imaging Syst. **3**(1) (2017)
16. Ting, D.S.W., et al.: Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. JAMA **318**(22), 2211–2223 (2017)
17. Tao, X., Zhang, H., Huang, X., Zhang, S., Metaxas, D.N.: Medical image computing and computer-assisted intervention—MICCAI 2016. Lecture Notes in Computer Science, vol. 9901. Springer, Cham
18. Yan, Z., et al.: Multi-instance deep learning: discover discriminative local anatomies for bodypart recognition. IEEE Trans. Med. Imaging **35**(5), 1332–1343 (2016)
19. Shimizu, R., Yanagawa, S., Monde, Y., Yamagishi, H., Hamada, M., Shimizu, T., Kuroda, T.: Deep learning application trial to lung cancer diagnosis for medical sensor systems. In: 2016 International SoC Design Conference (ISOCC) (2016)
20. Vieira, S., Pinaya, W.H.L., Mechelli, A.: Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: methods and applications. Neurosci. Biobehav. Rev. **74**, 58–75 (2017)
21. Avendi, M.R., Kheradvar, A., Jafarkhani, H.: A combined deep-learning and deformable-model approach to fully automatic segmentation of the left ventricle in cardiac MRI. Med. Image Anal. **30**, 108–119 (2016)
22. Cruz-Roa, A., et al.: Accurate and reproducible invasive breast cancer detection in whole slide images: a deep learning approach for quantifying tumor extent. Sci. Rep. **7**, 46450 (2017)
23. Levy, D., Jain, A.: Breast mass classification from mammograms using deep convolutional neural networks (2016). arXiv:1612.00542
24. Liu, S., Liu, S., Cai, W., Pujoi, S., Kikinis, R., Feng, D.: Early diagnosis of Alzheimer's disease with deep learning. In: 2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI), Beijing, pp. 1015–1018 (2014)
25. Rahhal, M.M., Bazi, Y., AlHichri, H., Alajlan, N., Melgani, F., Yager, R.R.: Deep learning approach for active classification of electrocardiogram signals. Inf. Sci. **345**, 340–354 (2016)
26. Chen, M., Shi, X., Zhang, Y., Wu, D., Guizani, M.: Deep feature learning for medical image analysis with convolutional autoencoder neural network. IEEE Trans. Big Data (2017)
27. Gao, L., Lin, S., Wong, T.Y.: Automatic feature learning to grade nuclear cataracts based on deep learning. IEEE Trans. Biomed. Eng. **62**(11), 2693–2701 (2015)
28. Albarqouni, S., Baur, C., Achilles, F., Belagiannis, V., Demirci, S., Navab, N.: AggNet: deep learning from crowds for mitosis detection in breast cancer histology images. IEEE Trans. Med. Imaging **35**(5), 1313–1321 (2016)
29. Chen, H., Yu, L., Dou, Q., Shi, L., Mok, V.C.T., Heng, P.A.: Automatic detection of cerebral microbleeds via deep learning based 3D feature representation. In: 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI), New York, NY, pp. 764–767 (2015)
30. Wang, D., Khosla, A., Gargeya, R., Irshad, H., Beck, A.H.: Deep learning for identifying metastatic breast cancer (2016). arXiv preprint arXiv:1606.05718
31. Han, Z., Wei, B., Zheng, Y., Yin, Y., Li, K., Li, S.: Breast cancer multi-classification from histopathological images with structured deep learning model. Sci. Rep. **7**, 4172 (2017)

32. Akkus, Z., Galimzianova, A., Hoogi, A., Rubin, D.L.: Deep learning for brain MRI segmentation: state of the art and future directions. J. Digit. Imaging **30**, 449–459 (2017)
33. Milletari, F., et al.: Hough-CNN: deep learning for segmentation of deep brain regions in MRI and ultrasound. Comput. Vis. Image Underst. **164**, 92–102 (2017)
34. Lee, C.S., Tyring, A.J., Deruyter, N.P., Wu, Y., Rokem, A., Lee, A.Y.: Deep-learning based, automated segmentation of macular edema in optical coherence tomography. Biomed. Opt. Express **8**(7), 3440 (2017)
35. Huynh, B.Q., Li, H., Giger, M.L.: Digital mammographic tumor classification using transfer learning from deep convolutional neural networks. J. Med. Imag. **3**(3), 034501 (2016)
36. Dhungel, N., Carneiro, G., Bradley, A.P.: Deep learning and structured prediction for the segmentation of mass in mammograms, medical image computing and computer-assisted intervention—MICCAI 2015. Lecture Notes in Computer Science, vol. 9349. Springer, Cham (2015)
37. Dhungel, N., Carneiro, G., Bradley A.P.: Automated mass detection in mammograms using cascaded deep learning and random forests. In: 2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA), 23–25 Nov 2015, Adelaide, SA, pp. 1–8 (2015)
38. Carneiro, G., Nascimento, J., Bradley A.P.: Unregistered multiview mammogram analysis with pre-trained deep learning models, medical image computing and computer-assisted intervention—MICCAI 2015. Lecture Notes in Computer Science, vol. 9351. Springer, Cham (2015)
39. Dalmis, M.U., Litjens, G., Holland, K., Setio, A., Mann, R., Karssemeijer, N., Gubern-Merida, A.: Using deep learning to segment breast and fibroglandular tissue in MRI volumes. Med. Phys. **44**, 533–546 (2017)
40. Isin, A., Direkoglu, C., Sah, M.: Review of MRI-based brain tumor image segmentation using deep learning methods. In: 12th International Conference on Application of Fuzzy Systems and Soft Computing, ICAFS 2016, 29–30 August 2016, Vienna, Austria, pp. 317–324 (2016)
41. Zhao, X., Wu, Y., Song, G., Li, Z., Zhang, Y., Fan, Y.: A deep learning model integrating FCNNs and CRFs for brain tumor segmentation. Med. Image Anal. **43**, 98–111 (2018)
42. Suk, H., Lee, S.W., Shen, D.: The Alzheimer's disease neuroimaging initiative, hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis. NeuroImage **101**, 569–582 (2014)
43. Lai, M.: Deep learning for medical image segmentation (2015). arXiv:1505.02000v1
44. Gulshan, V., et al.: Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA **316**(22), 2402–2410 (2016)
45. Abràmoff, M.D., Lou, Y., Erginay, A., Clarida, W., Amelon, R., Folk, J.C., Niemeijer, M.: Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning. Invest. Ophthalmol. Vis. Sci. **57**(13), 5200–5206 (2016)
46. Fu, H., Xu, Y., Wong, D.W.K., Liu, J.: DeepVessel: retinal vessel segmentation via deep learning and conditional random field. In: International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI 2016: Medical Image Computing and Computer-Assisted Intervention—MICCAI 2016, pp 132–139 (2016)
47. Liskowski, P., Krawiec, K.: Segmenting retinal blood vessels with deep neural networks. IEEE Trans. Med. Imaging **35**(11), 2369–2380 (2016)
48. Emad, O., Yassine, I.A., Fahmy, A.S.: Automatic localization of the left ventricle in cardiac MRI images using deep learning. In: 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan, pp. 683–686 (2015)
49. Kumar, D., Wong, A., Clausi, D.A.: Lung nodule classification using deep features in CT images. In: 2015 12th Conference on Computer and Robot Vision, pp. 133–138 (2015)

Deep Domain Adaptation for Regression

Ankita Singh and Shayok Chakraborty

Abstract Developing machine learning algorithms under the constraint of limited labeled data has attracted significant attention in the research community in recent years. Domain adaptation or transfer learning algorithms alleviate this challenge by transferring relevant knowledge from a source domain to induce a model for a related target domain, where labeled data are scarce. Further, deep learning algorithms are instrumental in learning informative feature representations from a given dataset and have replaced the need for hand-crafted features. In this chapter, we propose a novel framework, *DeepDAR*, for domain adaptation for regression applications, using deep convolutional neural networks (CNNs). We formulate a loss function relevant to the research task and exploit the gradient descent algorithm to optimize the loss and train the deep CNN. To the best of our knowledge, domain adaptation for regression applications using deep neural networks has not been explored in the literature. Our extensive empirical studies on two popular regression applications (age estimation and head pose estimation from images) depict the merit of our framework over competing baselines.

Keywords Domain adaptation · Deep learning · Regression

1 Introduction

Modern era is witnessing an unprecedented increase in various forms of digital data (such as images, videos, text etc.). Such data is highly unstructured, extremely diverse in its contents and has a high redundancy (for instance, frames in a video stream). Hand-labeling the data manually to learn a model necessitates significant human labor and time. This has motivated the development of machine learning algorithms under weak human supervision.

Domain Adaptation: Domain adaptation (DA) (or transfer learning) algorithms address the problem of learning with weak supervision by utilizing abundant labeled

A. Singh · S. Chakraborty (✉)
Department of Computer Science, Florida State University, Tallahassee, USA
e-mail: shayok@cs.fsu.edu

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_4

data in one domain to develop a model for a related domain of interest, where there is a paucity of labeled data [51]. The domain of interest is referred to as the *target* domain and the other domain is called the *source* domain. For instance, consider the problem of emotion recognition from facial images. Let us suppose we want to train a model to recognize emotions on Asian faces (target domain); however, our training data consists of images primarily from the Caucasian population (source domain), with a few labeled images from the target Asian population. As another example, consider the problem of human age estimation from facial images. Suppose, we have a large amount of labeled training data for male subjects (source domain); however, we are interested to develop a prediction model for female subjects, where labeled data are scarce. In these applications, acquiring large amounts of labeled target data is extremely challenging. Domain adaptation algorithms are immensely useful in such situations as they transfer pertinent information from the source to the target domain, to reduce the human annotation effort in learning a model in the target domain.

In a traditional supervised learning setting, we are given training samples $X = \{x_1, x_2, \ldots x_n\}$, together with class labels $Y = \{y_1, y_2, \ldots y_n\}$, and the objective is to learn $P(Y|X)$ from the given training data. The unseen test data is assumed to be derived from the same probability distribution as the training data; thus, a model trained on the training data is assumed to generalize well on the test set. In contrast, in domain adaptation, we are given data from a source domain D_S and a target domain D_T; the probability distributions generating the data in the two domains are different, which implies a difference in their joint probability distributions: $P_s(X, Y) \neq P_T(X, Y)$ [10]. In a typical DA setup, the source domain is assumed to contain plenty of labeled data. However, the target domain is assumed to contain no labeled data (unsupervised DA) or few labeled samples (semi-supervised DA). Since target domain samples are scarce, it is challenging to accurately compute $\hat{P}_T(X, Y)$. The main objective of DA is to approximate the distribution $\hat{P}_T(X, Y)$ using information from the source domain, to develop an accurate prediction model for the target domain. To this end, the source and target domains are assumed to be correlated, where $P_s(X) \neq P_T(X)$ but $P_s(Y|X) \approx P_T(X|Y)$; that is, the marginal distributions of the source and target are different, but their conditional distributions are the same [49]. Domain adaptation has found remarkable success in several computer vision applications, such as facial expression recognition [82, 86], object recognition [32, 62] and handwritten digits recognition [47, 63] among others. However, most of these algorithms have focused on the classification problem; the regression setting has received considerably less attention.

Deep Learning: In recent years, deep learning algorithms have depicted remarkable performance in a variety of applications. One of the fundamental challenges in computer vision is to extract a discriminating set of features from images and videos. Vision researchers have conventionally used hand-crafted feature extraction techniques, which are data-agnostic and task-agnostic. Common examples include: (i) the Histogram of Oriented Gradients (HoG) [16], (ii) GIST features [48] and (iii) the speeded up robust features (SURF) [5]. In contrast, deep learning algorithms automatically learn the best set of features for a given dataset; the features learned are task and data specific and thus depict much better performance than hand-engineered features. Convolutional Neural Networks (CNNs) are most commonly used in computer vision research. A typical CNN contains the following types of layers: (i) input layer to hold the raw values of the data; (ii) convolution layer to compute the output of neurons using convolution filters over local regions within the input; (iii) $ReLU$ layer, which applies an element-wise activation function, such as the $max(0, x)$, thresholding at zero; (iv) pool layer to perform a down sampling operation along the spatial dimensions; and (v) fully-connected layer to compute the class probabilities. Through multiple layers of non-linearities, CNNs can extract highly discriminating features for a given dataset. Common examples of CNN architectures include: (i) the $AlexNet$, which contains 5 convolutional layers and 3 fully connected layers [36]; (ii) the $GoogLeNet$, which contains 22 layers and much lesser number of parameters than the $AlexNet$ [73]; and (iii) the $ResNet$, which contains 152 layers [31].

Deep learning has advanced the state-of-the-art on a variety of applications, including image recognition [31, 36, 73], image segmentation [4, 14], text mining [37, 84], and medical diagnosis [40, 43], among others. The unparalleled success of deep learning has motivated vision researchers to exploit its capabilities to address the challenging problem of domain adaptation. In conventional shallow transfer learning approaches, the hand-crafted features are first extracted from the dataset, and then a model is developed to align the source and the target domains, for the fixed set of features [49, 71, 78]. In contrast, deep domain adaptation algorithms directly learn transferable feature representations for the source and target data in question and have depicted much improved empirical performance [24, 44, 77].

All the deep learning-based algorithms for domain adaptation have been developed for classification problems only; the regression setting has not been explored. Motivated by their success in the classification setting, in this paper, we attempt to address the problem of domain adaptation in the regression setting, using deep neural networks. Our novel framework, **Deep** **D**omain **A**daptation for **R**egression ($DeepDAR$), is based on training the deep learning model to optimize a unique loss function with the following components: (i) a supervised loss for the labeled source data and labeled target data (if available), which ensures that the trained model is consistent with the given labeled data; (ii) a loss based on Maximum Mean Discrepancy (MMD), to minimize the probability distribution difference between the source and target domains and learn transferable features accordingly; and (iii) a semi-supervised loss for unlabeled target data, based on the Graph Laplacian, which enforces the underlying model to smoothly fit the data. The framework is flexible and can be easily adapted to both unsupervised and semi-supervised DA and for multivariate regression, where more than one output variable needs to be predicted. This research is the first of its kind in exploiting the feature learning capabilities of deep convolutional neural networks (CNNs) to address the challenging problem of domain adaptation for regression applications. Although validated on vision data in this paper, $DeepDAR$ is a generic framework to learn a discriminating set of features to address the domain disparity in any regression setting.

2 Related Work

In this section, we present a survey of related work. Since the primary focus of this research is domain adaptation using deep learning, we organize this survey into two parts. In the first part, we present literature on domain adaptation using deep learning models. In the second part, we focus on existing work on domain adaptation for regression applications.

Domain Adaptation using Deep Learning: Domain adaptation (DA) is a well-studied problem in machine learning [51]. Before the advent of deep learning [36], researchers primarily relied on hand-crafted features for DA [3, 10, 18, 22, 28, 50, 61]. DA techniques based on deep learning have outperformed their non-deep counterparts due to the highly informative feature representations learned by the deep models.

In the Deep Domain Confusion (DDC) algorithm proposed by Tzeng et al. [76], domain invariant features were learnt in the *fc8* layer using an AlexNet [36] model. The disparity between the source and target domains was quantified using the Maximum Mean Discrepancy (MMD), which was appended as a term in the loss function used to train the network. Along similar lines, Tzeng et al. [74] augmented a domain classification loss term with the MMD, to learn discriminative features for the source and target data. Long et al. [44] proposed the Deep Adaptation Networks (DAN) model where the MMD loss was applied in all the fully connected layers (*fc6, fc7 and fc8*) of the AlexNet, with promising empirical performance. The Residual Transfer Network (RTN) architecture, proposed by Long et al. [45], incorporated a residual layer in the network and used MMD to address domain disparity. DA has also been used to learn informative hash codes for the source and target data, while addressing the probability distribution difference between them [77]. Other related techniques include a residual parameter transfer framework for deep domain adaptation [60], studying domain shift in action videos [33] and a zero-shot framework for deep DA [53]. Le et al. [38] recently conducted a rigorous theoretical analysis of the discrepancy between the source and target domain data and its relation to the reduction in classification errors. Multi-modal DA has also been explored using deep neural networks [56].

The Generative Adversarial Networks (GANs) proposed by Goodfellow et al. [27] is one of the hallmarks of deep learning research. GANs are generative models capable of generating data (text, images, audio, etc.) according to a specified distribution *P(X)*. Several recent techniques apply adversarial training for domain adaptation [75]. The Domain Adversarial Neural Network (DANN) architecture, proposed by Ganin et al. [24] incorporates a domain classifier, whose gradient is reversed when learning the feature extractor weights. Liu and Tuzel [42] implemented a Coupled Generative Adversarial Network (CoGAN) model, which shares weights at different layers of the GAN to train a coupled network. Further, CoGAN has been combined with Variational Autoencoder (VAE) [35] to develop an image translation network [41]; the network was utilized to convert target images to source images which could be classified efficiently using a source classifier. Data augmentation for DA was explored by Volpi et al. [79], where source data was augmented using a feature generator S, which was used to learn domain invariant features through a minimax framework. Concepts from Wasserstein GAN [2] have also been used for domain adaptation [64]. The adversarial methods based on GANs depict commendable performance; however, these networks cannot be fine-tuned and need to be trained from scratch, necessitating large amounts of labeled data.

Domain Adaptation for Regression: Contrary to the classification setting, domain adaptation in regression has received much less attention. A category of algorithms has focused on importance weighting (estimating the weights of the training samples to address the probability distribution difference between the source and the target) for DA. Yamada et al. [81] proposed a semi-supervised DA framework for structured regression where each training sample was assigned a weight based on its similarity to the test samples in a high dimensional space. Garcke and Vanck [25] proposed an algorithm for inductive transfer learning, where the importance weight function was defined as:

$$w(x, y) = \frac{P_T(x, y)}{P_S(x, y)} \tag{1}$$

Using the function *w(x,y)*, an appropriate weight was assigned to each source sample so as to minimize the probability distribution difference between the source and the target. Nearest neighbor-based importance weighting schemes have also been devised for DA in regression. Guan et al. [29] introduced the k-NN based weighting scheme where the importance of each training sample was computed based on its distance to the k closest neighbors in the test set. The authors also proposed a clustering based weighting strategy where the importance of each training data x_i was computed as [29]:

$$w(x_{tr}^i) = \frac{|Clus_{te}(x_{tr}^i)|}{|Clus_{tr}(x_{tr}^i)|} \tag{2}$$

where $w(x_{tr}^i)$ denotes the weightage of the training sample x_{tr}^i and $|Clus_{tr}(x_{tr}^i)|$ and $|Clus_{te}(x_{tr}^i)|$ denote the number of training and test samples respectively, in the same cluster as x_{tr}^i. However, the weight computation is mostly based on heuristic measures in most of these algorithms. Boosting based algorithms have also been explored for domain adaptation in regression. Pardoe and Stone [52] introduced the boosted transfer stacking and the two stage TrAdaboost.R2 algorithms, which are extensions of their counterparts in the classification setting. Bhattarai et al. [7] proposed a DA framework for age estimation, which learned a projection mapping, along with a regressor in the projected space through a single framework. The problem was posed as learning the projection matrix L and the regressor w jointly through a single optimization framework, which was solved using stochastic gradient descent. Cao et al. [11] proposed an adaptive transfer Gaussian Process (AT-GP) framework for domain adaptation, which was based on the following two conditions: (i) the shared knowledge between tasks should be transferable as much as possible; and

(ii) negative transfer should be avoided when these tasks are unrelated. Cortes and Mohri conducted an extensive analysis of the theoretical properties of DA algorithms in the regression setting [15]. The Transfer Component Analysis (TCA) algorithm [49], projects the source and target data onto a latent subspace such that the domain disparity is minimized in the projected space; machine learning algorithms are then trained on the new representations in the latent subspace.

As evident from this survey, all the DA algorithms for regression applications work on hand-crafted features. Further, most of the algorithms work only in the presence of labeled data in the target domain and are unable to utilize unlabeled target data. In this paper, we introduce *DeepDAR*, a novel algorithm to address these challenges. We exploit the power of deep CNNs to learn a discriminating feature set for the domain adaptation task. Our framework can utilize unlabeled data in the target domain and outperforms competing baselines on a variety of applications. Our framework is detailed in the following section.

3 Proposed Framework

We are given data from two domains: source and target. Typically, the data in the source domain are completely labeled: $D_S = \{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{i=1}^{\boldsymbol{n_s}}$. In the target domain, a small fraction of the data is labeled: $\boldsymbol{D_T^L} = \{\boldsymbol{x_j}, \boldsymbol{y_j}\}_{j=1}^{\boldsymbol{n_T^L}}$ and a major portion is unlabeled: $\boldsymbol{D_T^U} = \{\boldsymbol{x_j}\}_{j=1}^{\boldsymbol{n_T^U}}$. For a regression application, the labels y are all real numbers. As mentioned previously, the goal of domain adaptation is to induce a learning model in the target domain by intelligently using both source and target domain data, in the presence of a disparity between the two domains. Given the labeled source $\boldsymbol{D_S}$, labeled target $\mathbf{D}_{\boldsymbol{T}}^{\boldsymbol{L}}$ and unlabeled target $\boldsymbol{D_T^U}$ as training data, we train a deep neural network to predict unseen test data in the target domain. Instead of using an off-the-shelf network trained on a different dataset, for a different application, we propose to formulate a novel loss function specific to the application in question and train the network to optimize that loss. Our network will then get specifically tailored to our application and can potentially depict improved learning performance. Our loss function consists of three components: (i) supervised loss on the labeled data, which enforces the network to incur minimal prediction error on the labeled source and target (if any) samples; (ii) an MMD loss between the source and the target, which attempts to address the disparity between the two domains and learn features accordingly; and (iii) a semi-supervised loss on the unlabeled target data, which encourages the prediction function to smoothly fit the data. These are detailed below.

3.1 Supervised Loss on Labeled Data

The goal of the supervised loss term is to encourage the trained deep network to furnish accurate predictions on the labeled data. In the regression setting, this is quantified by the mean squared error (MSE) between the ground truth and predicted labels. Let $D^L = D_S \cup D_T^L = \{x_1, x_2, \ldots x_l\}$ be the labeled source and target data (if available), with corresponding labels $\{y_1, y_2, \ldots y_l\}$ The supervised loss on the labeled data is computed as (in the absence of labeled target data, this term is computed only on the source data):

$$L_{Sup}(D^L) = \frac{1}{l}\sum_{i=1}^{l}(y_i - \hat{y}_i)^2 \tag{3}$$

where $\hat{y}_i$ is the predicted output for y_i.

3.2 MMD Loss Between Source and Target Data

The objective of this term is to quantify the distance between the source and target distributions; minimizing this distance will reduce the domain disparity between the source and the target. The concept of Maximum Mean Discrepancy (MMD) was proposed by Borgwardt et al. [9] as a metric to compare distributions based on the Reproducing Kernel Hilbert Space (RKHS) [67, 68]. Let $A = \{a_1, a_2, \ldots a_{n1}\}$ and $B = \{b_1, b_2, \ldots b_{n2}\}$ be random variables with distributions P and Q respectively. The MMD between P and Q is empirically computed from the given samples as:

$$MMD(P, Q) = \|\frac{1}{n1}\sum_{i=1}^{n1}\emptyset(a_i) - \frac{1}{n2}\sum_{i=1}^{n2}\emptyset(b_i)\|_{\mathcal{H}}^2 \tag{4}$$

where $\mathcal{H}$ is a universal RKHS [69] and $\varphi : X \rightarrow \mathcal{H}$.

The MMD metric has been used to quantify the discrepancy between the source and target domains in domain adaptation research [44, 77]. Given the source data DS and the target data $D_T = D_T^L \cup D_T^U$, the MMD loss between the source and the target is computed as follows:

$$L_{MMD}(D_S, D_T) = \|\mathbb{E}[\varphi(D_S)] - \mathbb{E}[\varphi(D_T)]\|_{\mathcal{H}}^2 \tag{5}$$

where E denotes the expectation function and $\varphi : X \rightarrow \mathcal{H}$ is the kernel. We append this as a term in our loss function; minimizing this term ensures that the feature representations are learned in such a way that the difference between the source and target probability distributions is minimized.

3.3 Semi-supervised Loss for Unlabeled Target Data

While most DA algorithms for regression work only in the presence of labeled target data, our method is capable of leveraging abundant unlabeled target training data to induce a robust model. In a classification setting, the unlabeled data is usually exploited through some uncertainty metric, such as Shannon's entropy. The entropy of each unlabeled sample is computed from the probability distribution of the network outputs on that sample and the total entropy on the unlabeled set is included as a term in the overall loss function [57, 58]. Minimizing this term ensures that the network is trained such that it furnishes high confidence (low entropy) predictions on the unlabeled set. However, computing the prediction uncertainty in a regression application is a challenge, as entropy does not have an exact analogue in the regression setting. We therefore impose a condition which ensures that the underlying prediction function fits the labeled and unlabeled data smoothly and express this in terms of the unlabeled target data.

We exploit ideas from graph transduction, which has shown remarkable success in semi-supervised learning, to derive a semi-supervised loss term for the unlabeled target data. As before, let $D^L = D_S \cup D_T^L = \{x_1, x_2, \dots x_l\}$ be the labeled source and target data (if available), with corresponding labels $\{y_1, y_2, \dots y_l\}$. Let $D_T^U = \{x_{l+1}, \dots x_n\}$ be the unlabeled target data. Let $f = [f(x_1), f(x_2), \dots f(x_n)]$ denote the outputs of a regressor f (that we are trying to learn) on the labeled and unlabeled data. Manifold regularization-based techniques attempt to construct a smoothness term based on the intrinsic geometry of the data, which ensures that if two samples xi and x_j are close to each other, then $f(x_i)$ should be similar to $f(x_j)$ [12]. Each sample x_i is treated as a node in an undirected graph and the edge between samples x_i and x_j is denoted by e_{ij}. Let w_{ij} be the weight of the edge e_{ij}. A kernel function is typically used to compute the weights: $w_{ij} = k(x_i, x_j)$, with the Gaussian kernel being a popular choice:

$$w_{ij} = exp(\frac{-||xi - xj||^2}{2\sigma^2}) \tag{6}$$

The edge weight matrix is given as $W = \{w_{ij}\}$. D is a diagonal matrix given by $D_{ii} = \sum_{j=1}^{n} W_{ij}$. The graph Laplacian matrix is then computed as $L = D - W$ [12]. Using the Laplacian matrix, the smoothness of the prediction function f can be computed in terms of a quadratic form of the Graph Laplacian [17, 85]:

$$f^T L f = \frac{1}{2} \sum_{i,j} w_{ij} (f(i) - (f(j))^2 \tag{7}$$

Intuitively, Eq. (7) penalizes the difference between $f(x_i)$ and $f(x_j)$ more when w_{ij} is large; it thus enforces the smoothness assumption by encouraging strongly connected vertices (with a large weight on the edge between them) to have similar values. The smaller the quadratic form in Eq. (7), smoother is the prediction function f. The smoothness constraint has been exploited in a variety of graph-based learning problems [66, 85]. We append this as a term in our loss function to leverage unlabeled target data in training the deep neural network:

$$L_{SemiSup}(D^L, D_T^U) = f^T L f = \frac{1}{2}\sum_{i,j} w_{ij}(f(i) - (f(j))^2 \tag{8}$$

The overall loss function is given as a weighted summation of the supervised, MMD and semi-supervised loss terms:

$$L_{total} = L_{Sup}(D^L) + \lambda_1 L_{MMd}(D_S, D_T) + \lambda_2 L_{SemiSup}(D^L, D_T^U) \tag{9}$$

where λ_1 and λ_2 are the importance weights. The network was trained using the back-propagation algorithm, where the gradient of the objective was computed using the differentiation methods available in Python.

4 Experiments and Results

In this section, we present an empirical analysis of our framework against relevant baselines. We studied the effects of the size of the labeled source set, labeled target set and kernel functions on the predictive performance. These are presented in the following sections.

4.1 Datasets and Feature Extraction

We validated the performance of *DeepDAR* on two popular vision-based regression applications—age estimation and head pose estimation from images. Both of these are well-studied problems in the vision research community [8, 23, 26, 30]. Note that our objective in this research was to study the performance of deep learning based domain adaptation algorithms, and not to outperform the state-of-the-art error rates on these datasets. We therefore used random subsets of images from each dataset and did not replicate the exact train/test splits used in previous research, where the objective was to achieve the lowest prediction error rates [19].

Age Estimation: The IMDB-WIKI dataset contains face images with gender and age labels of celebrities from the IMDB and Wikipedia databases [59]. Figure 1 depicts sample images from IMDB-WIKI.

Head Pose Estimation: The Biwi Kinect Head Pose dataset contains pose images of subjects recorded using a Kinect [20]. The ground truth for each image is provided as the 3D location of the head and its yaw, pitch and roll angles. The Queen Mary

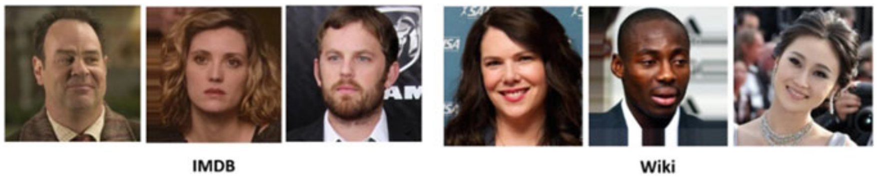


Fig. 1 Sample images from the IMDB and Wiki datasets

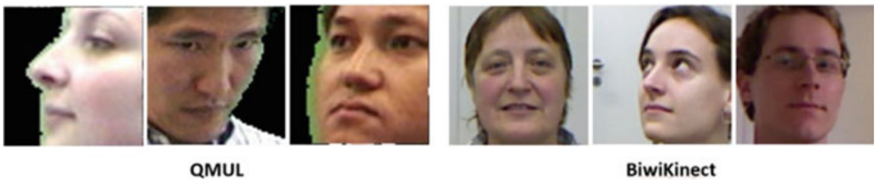


Fig. 2 Sample images from the Biwi Kinect and QMUL datasets

University of London Multiview Face Dataset (QMUL) also consists of face images of subjects exhibiting a wide range of poses [65]. We considered only the yaw angles as the variable to be predicted in this work. Figure 2 depicts sample images from these datasets.

Inspired by previous research on domain adaptation for regression applications, we used the male and female subjects in each dataset to form the two domains [7, 29]. Further, for the IMDB and Wiki datasets, we also conducted a cross-dataset experiment, where subjects in one dataset were taken as the source and those in the other, as the target.

Feature Extraction: As mentioned previously, ours is the first framework to address the problem of DA in regression using deep CNNs. The comparison baselines (detailed below) work only on hand-crafted features. For fair comparison, we extracted deep features using the pre-trained VGG-F model [13] from each image and passed them as inputs to the baseline methods.

4.2 Comparison Baselines

We used the following algorithms as comparison baselines in our work:

No Source: Here, the source data was not used at all. We merely trained a regression model on the labeled target data and predicted on the test data.

No DA: In this baseline, we used the source data without applying any domain adaptation algorithm on it. A regression model was trained on the labeled source and the labeled target and tested on the test set.

TCA: The Transfer Component Analysis algorithm proposed by Pan et al. [49], which attempts to find a common latent representation for both the source and target data, that preserves the data configuration of the two domains after transformation; machine learning algorithms are then trained on the new representations in the latent subspace.

R2: The boosting based regression transfer learning algorithm, TrAdaBoost.R2 [52].

JL: The Joint Learning approach proposed by Bhattarai et al. [7], which uses metric learning to learn a low dimensional projection (which aligns the features from the source and target domains) along with a regressor in the projection space to predict from the domain aligned features.

DITL: The Direct Inductive Transfer Learning method proposed by Garcke and Vanck [25], which uses an importance weighting scheme to assign a weight to all the source samples in order to train a regression model.

4.3 Deep Network Architecture

We used a neural network of 12 convolutional layers. The input images were scaled to 128×128 and the pixels were normalized to remove any illumination effects and for faster convergence [39]. The filter size in the convolutional layers was set to the minimum size of 3×3 that retains the up, down, left, right and center notion in images. Rectified linear unit (ReLU) was used as the activation function in all the hidden layers. The weights were initialized with Xavier normal initializer while bias was set to zero. Some of the convolutional layers were followed by max-pooling layers to carry out the spatial pooling over a 2×2 window with a stride of 2. In total, 7 max-pooling layers were used. The number of filters in the 12 convolutional hidden layers varied from 32 to 512 in an incremental order of 2. The stack of convolutional layers was followed by one fully connected hidden layer with 512 units and an output *sigmoid* layer for predicting the continuous output value. The mini-batch size was fixed to 16 samples. Each mini-batch consists of 8 samples from the labeled source set, 2 from the labeled target set and 6 from the unlabeled target set. The network was tested with the Adam, RMSProp and Adadelta optimizers with different datasets and learning rates. RMSProp optimizer with a learning rate of $10-4$ was found to be the optimum for this study. For regularization, dropouts after 4 max-pooling layers with a ratio of 0.25 was used. The network was trained for 50 epochs. The network architecture is depicted in Fig. 3.

Label Normalization: Both the age estimation datasets (IMDB and Wiki) as well as the head pose estimation datasets (Biwi Kinect and QMUL) had a large spread in the values of the output variables. The minimum value of age for the IMDB dataset, for instance, is 6 years and the maximum value is 101 years. For the QMUL head pose dataset, the minimum pose angle is 10° and the maximum is 180°. Such a wide variation in the values of the target variable may produce error gradients of high magnitude; this may result in drastic variations of the weight values from one epoch to another, making the learning process unstable [1, 55]. As suggested by [55], we scaled the output variables to the range [0–1] and used the scaled values in all our

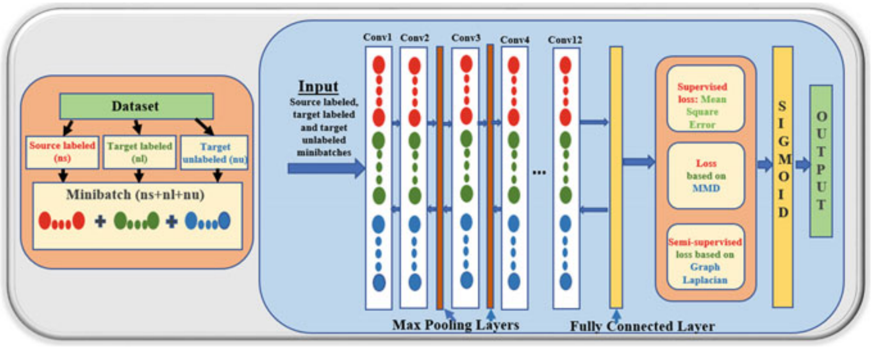


Fig. 3 Architecture of the deep network used in our study. Best viewed in color

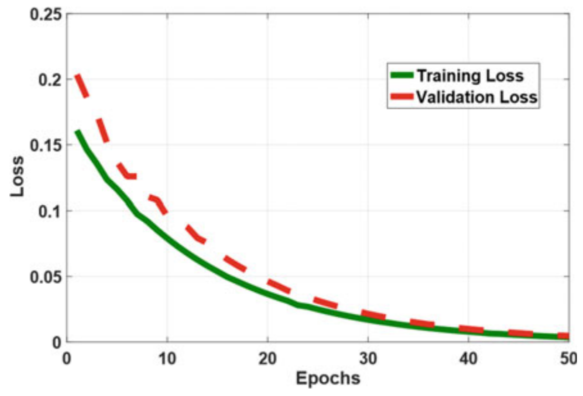


Fig. 4 Plot of the training and validation errors against the number of epochs. The IMDB dataset is taken as the source and the Wiki dataset as the target. Best viewed in color

empirical studies. For consistency, the scaled label values were also used to train the baseline methods.

Figure 4 depicts a sample plot of the training and validation errors against the number of epochs, while training our deep neural network for the experiment where the IMDB dataset is taken as the source and the Wiki dataset as the target. We note a steady decline of the errors on both the training and validation sets, as the training epochs increase. A similar plot was obtained while training our deep network for the other datasets, which shows that our deep model is robust to overfitting issues.

4.4 *Experimental Setup*

In each experiment, we have a source set and a target set. The target set was divided into three parts: a labeled training set, an unlabeled training set and a test set. The number of images in each set, for each experiment are detailed in Table 1. In a real-world application, the number of labeled target data is much less than the number of

Table 1 Number of sources, labeled and unlabeled target and test samples for each dataset used in our experiments. **S**: Number of source samples, **LT**: Number of labeled target samples, **UT**: Number of unlabeled target samples, **T**: Number of test samples, **M**: Male subjects, **F**: Female subjects

Source	Target	S	LT	UT	T
Wiki (F)	Wiki (M)	9,605	3,500	8,000	4,500
Wiki (M)	Wiki (F)	7,000	2,250	4,750	2,250
Wiki (M and F)	IMDB (M and F)	38,104	10,000	25,000	15,000
IMDB (M and F)	Wiki (M and F)	20,000	8,000	18,000	8,000
Biwi (F)	Biwi (M)	3,447	1,000	2,500	2,131
Biwi (M)	Biwi (F)	5,624	700	1,500	1,247
QMUL (F)	QMUL (M)	1,505	896	1,349	1,215
QMUL (M)	QMUL (F)	2,500	305	600	600

unlabeled target data, as the fundamental premise of DA is the paucity of labeled data in the target domain. The sizes of the sets were selected to appropriately mimic this real-world situation, as depicted in Table 1. The objective was to utilize the labeled source data, labeled and unlabeled target training data to develop a model for the target test data. The parameters λ_1 and λ_2 in Eq. (9) were both taken as 100 based on preliminary studies. A gaussian kernel was used to compute the MMD in Eq. (5).

4.5 Performance Metrics

We used the mean squared error (MSE) and the mean absolute error (MAE) as the performance metrics in our experiments, both of which are extensively used to quantify the performance of regression algorithms.

4.6 Performance Analysis

The performance of the proposed method and the comparison baselines are reported in Tables 2, 3, 4, 5, 6, 7, 8 and 9.

The *TCA* and *R2* methods depict poor performance on most of the datasets. The *JL* method depicts impressive performance on some of the datasets, but is not consistent across datasets in its performance. The *No Source* method depicts the best performance on 3 experiments (in terms of MSE), showing that training a model merely on the target data can sometimes produce good performance. However, its performance on the other experiments is much worse than the best performing method. *No DA* produces the best performance on 2 experiments (in terms of MSE), showing its usefulness in datasets where the disparity between the source and the target domains is not very strong. But for most of the other datasets, it incurs much higher

Table 2 MSE and MAE analysis on the Wiki dataset

Method	MSE	MAE
No source	0.76	0.24
No DA	0.47	0.20
TCA	0.15	0.25
R2	0.90	1.2
JL	0.18	0.32
DITL	**0.02**	0.11
DeepDAR	**0.02**	**0.10**

Source Female subjects, Target: Male subjects. Best results are shown in bold

Table 3 MSE and MAE analysis on the Wiki dataset

Method	MSE	MAE
No source	**0.02**	**0.11**
No DA	**0.02**	0.12
TCA	1.26	1.45
R2	1.84	1.35
JL	0.14	0.35
DITL	**0.02**	0.13
DeepDAR	**0.02**	**0.11**

Source Male subjects, Target: Female subjects. Best results are shown in bold

Table 4 MSE and MAE analysis

Method	MSE	MAE
No source	**0.01**	0.10
No DA	**0.01**	0.10
TCA	0.76	0.25
R2	2.06	1.43
JL	0.43	0.20
DITL	0.02	0.11
DeepDAR	**0.01**	**0.09**

Source Wiki, Target: IMDB. Best results are shown in bold

errors compared to the best performing method. The *DITL* algorithm depicts the best performance among all the baselines considered. *DeepDAR* depicts the best performance in 7 out of the 8 experiments in terms of MSE, and 6 out of the 8 experiments in terms of MAE. This consistent performance can be attributed to the discriminative features learned by our method through our novel loss function (Eq. (9)); the features learned by our network are thus specially tailored to the application in question, which results in much improved performance across a variety of applications and

Table 5 MSE and MAE analysis

Method	MSE	MAE
No source	0.03	0.16
No DA	0.03	0.17
TCA	1.21	1.00
R2	1.88	1.36
JL	0.15	0.36
DITL	**0.02**	0.13
DeepDAR	**0.02**	**0.11**

Source IMDB, Target: Wiki. Best results are shown in bold

Table 6 MSE and MAE analysis on the Biwi Kinect dataset

Method	MSE	MAE
No source	0.03	0.13
No DA	0.03	0.17
TCA	0.74	0.24
R2	1.35	1.15
JL	0.04	0.15
DITL	0.02	0.11
DeepDAR	**0.0004**	**0.002**

Source Female subjects, Target: Male subjects. Best results are shown in bold

Table 7 MSE and MAE analysis on the Biwi Kinect dataset

Method	MSE	MAE
No source	0.05	0.18
No DA	0.03	0.16
TCA	0.98	1.05
R2	1.22	1.51
JL	0.07	0.22
DITL	0.03	0.14
DeepDAR	**0.0004**	**0.002**

Source: Male subjects, Target: Female subjects. Best results are shown in bold

experiments. The results strongly corroborate the promise and potential of *DeepDAR* for regression-based domain adaptation applications.

Table 8 MSE and MAE analysis on the QMUL dataset

Method	MSE	MAE
No source	0.21	0.46
No DA	0.43	0.66
TCA	0.29	**0.11**
R2	2.19	1.44
JL	0.30	0.44
DITL	**0.09**	0.24
DeepDAR	**0.09**	0.13

Source Female subjects, Target: Male subjects. Best results are shown in bold

Table 9 MSE and MAE analysis on the QMUL dataset

Method	MSE	MAE
No source	**0.09**	**0.26**
No DA	0.31	0.27
TCA	3.65	2.18
R2	0.34	0.50
JL	0.34	0.49
DITL	0.14	0.57
DeepDAR	0.17	0.42

Source Male subjects, Target: Female subjects. Best results are shown in bold

4.7 *Effect of Labeled Target Data*

In this experiment, we study the effect of labeled target data on the predictive performance. The results for the experiment where the male subjects in the Wiki dataset are taken as the source and the female subjects as the target, are depicted in Figs. 5 and 6 respectively (we plot the results of only the four top performing algorithms and the total error instead of the mean error, to better observe the pattern in the given scale).

We conducted experiments with 100, 50, 25 and 12.5% of the labeled target data; the percentages are with respect to the size of the original labeled target data for this experiment, as depicted in Table 1. The errors show an increasing trend with decreasing percentage of labeled target training data, which corroborates our intuition. More importantly, we note that the proposed method, *DeepDAR*, depicts the best performance consistently across all sizes of the labeled target data.

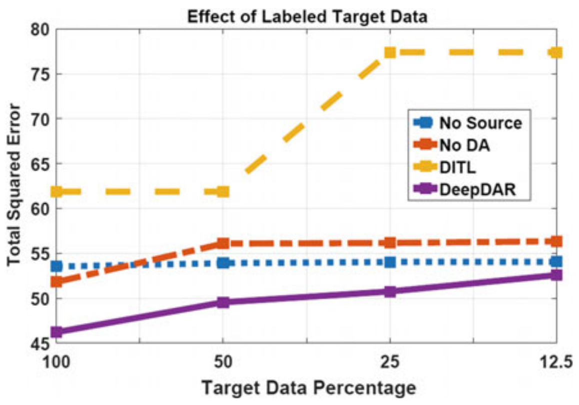


Fig. 5 Study of labeled target data: total squared error results. *Source* Male subjects in Wiki, Target: Female subjects in Wiki. Best viewed in color

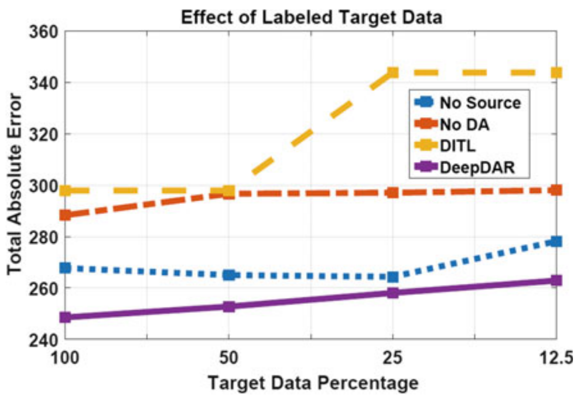


Fig. 6 Study of labeled target data: total absolute error results. *Source* Male subjects in Wiki, Target: Female subjects in Wiki. Best viewed in color

4.8 Effect of Labeled Source Data

The effect of labeled source data on learning performance is studied in this section. Figures 7 and 8 respectively depict the results of the experiment where the male subjects in Wiki are taken as the source and the female subjects in Wiki as the target (as before, we plot the results of only the four top performing algorithms and the total errors).

The *No Source* method uses only the labeled target samples to induce the model and thus, its performance remains unchanged with decreasing percentage of labeled source data. For the other methods, the error mostly shows an increasing pattern with decreasing percentage of labeled source data. *DeepDAR* once again depicts impressive performance across varied percentages of source samples. These results show the robustness of our approach to varying sizes of the labeled source and target training data.

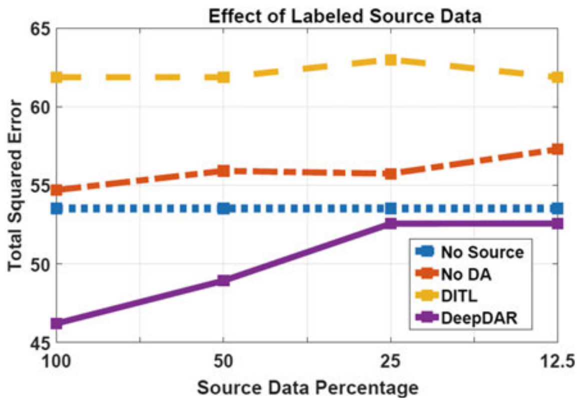


Fig. 7 Study of labeled source data: total squared error results. *Source* Male subjects in Wiki, Target: Female subjects in Wiki. Best viewed in color

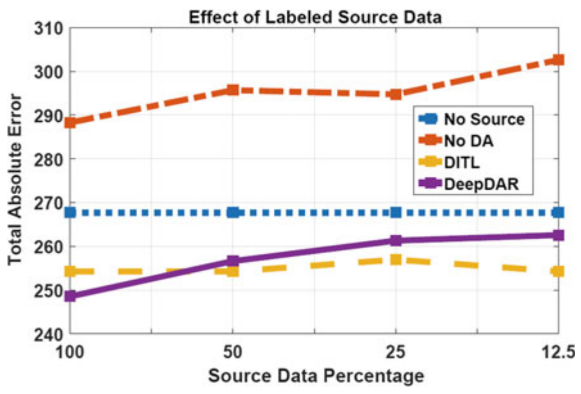


Fig. 8 Study of labeled source data: total absolute error results. *Source* Male subjects in Wiki, Target: Female subjects in Wiki. Best viewed in color

4.9 Effect of Kernel Function

In this section, we study the effect of the kernel function used to compute the MMD-based loss term in Eq. (5).

Figure 9 depicts the MSE and MAE results where the female subjects in QMUL are taken as the source and the male subjects as the target (we drop the *TCA* and *R2* methods, as they depict high errors). *DeepDAR*, with both the Gaussian and polynomial kernels, depicts the best performance among all the baselines. Analogous results are shown in Fig. 10, where the IMDB dataset is taken as the source and the Wiki as the target. As evident from the figure, the polynomial kernel depicts even better performance than the Gaussian kernel in terms of both MSE and MAE for this experiment. This shows the generalizability of our method across different kernel functions.

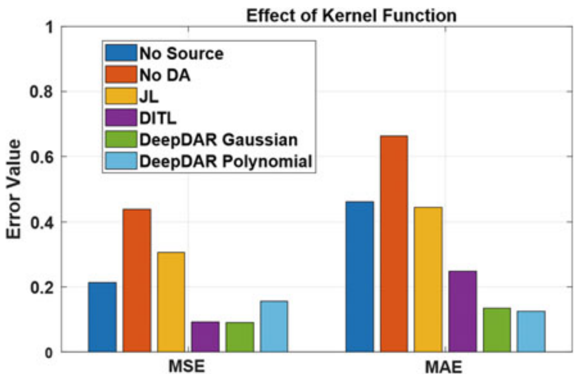


Fig. 9 Study of the Kernel Function. *Source* Female subjects in QMUL, Target: Male subjects in QMUL. Best viewed in color

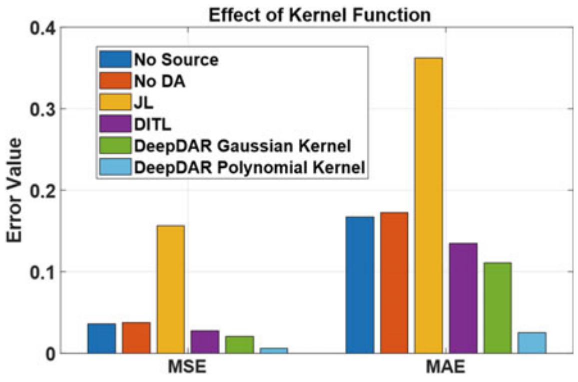


Fig. 10 Study of the Kernel Function. *Source* IMDB, Target: Wiki. Best viewed in color

4.10 Feature Analysis

In this section, we performed a comparative analysis of the features learned using the proposed method *DeepDAR* against the deep features used for the baseline methods [13]. Figure 11 depicts the $\mathcal{A}$-distance between the source and target domains for the two set of features, for two of our experiments (male subjects as source and female subjects as target for the Wiki and Biwi Kinect datasets).

The $\mathcal{A}$-distance between two domains is a measure of discrepancy between the two domains [6]. It is approximated as $2(1-2\in)$, where $\in$ is the generalization error of a binary classifier trained to distinguish samples from the two domains. A high value of the error (low value of $\mathcal{A}$-distance) implies that a binary classifier is unable to differentiate source and target samples; this means that the source and the target features have been well-aligned, so as to make the two domains almost indistinguishable. We used an SVM trained on a Gaussian kernel to estimate $\in$ (about 8,000 samples were taken from the source and target domains in each experiment and split equally into training and test sets). As evident from Fig. 11, the features learned using *DeepDAR* produce a higher error or lower discrepancy between the

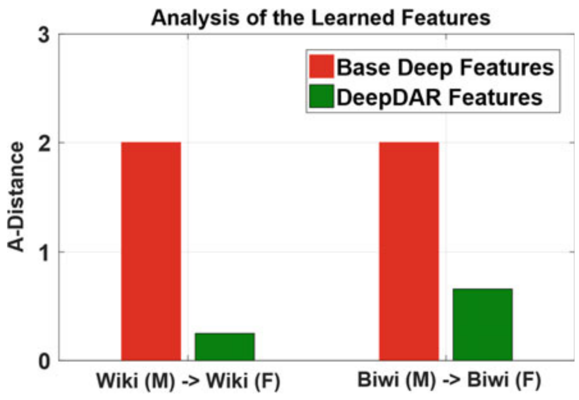


Fig. 11 Feature Analysis using $\mathcal{A}$-distance. *DeepDAR* produces lower domain disparity between the source and the target. The notation A $\rightarrow$ B implies A is the source domain and B is the target domain. Best viewed in color

source and the target, as compared to the deep features used for the baseline methods. Thus, the features learned using our method can address the domain disparity much better than features which are agnostic to the application. This shows the usefulness of our algorithm to learn domain invariant features using deep neural networks.

5 Conclusion and Future Work

We introduced *DeepDAR*—a novel deep learning framework to address the challenging problem of domain adaptation for regression applications. We formulated a loss function relevant to the domain adaptation problem and trained a deep convolutional neural network by optimizing the loss. Contrary to most existing DA algorithms for regression, *DeepDAR* can utilize the presence of unlabeled data in the target domain in learning a discriminating set of features. To the best of our knowledge, domain adaptation for regression using deep neural networks has not been explored in the literature. We validated the performance of our framework on image-based age estimation and head pose estimation. *DeepDAR* outperformed competing baselines in terms of the mean absolute and mean squared prediction errors and also the A-distance, depicting the discrepancy between the two domains.

In our future research, we plan to study the performance of *DeepDAR* on other regression applications. For instance, due to the unprecedented increase in the popularity of online platforms such as YouTube, Facebook, Twitter, Flickr etc., image/video popularity prediction (where the goal is to predict the popularity score of an image before it is uploaded online) has recently gained popularity in the research community [34, 54, 72]. However, all the popularity prediction algorithms leverage data from a particular website and train a model for the same website; these models fail to take advantage of the abundant data that is available in other photo-sharing websites. For instance, the images and the corresponding popularity scores from Instagram can be potentially useful in developing a better prediction model for

Flickr. Domain adaptation can be used to reduce the domain disparity between the two websites in order to address this challenge. The performance of *DeepDAR* on this application will be explored in our future work.

We also plan to study the performance of our framework for multivariate regression applications (with more than one output variables). For instance, in the Biwi Kinect head pose dataset, the ground truth pose values of each image are given as a triplet of yaw, pitch and roll angles [21] (we have only considered the yaw in this research). As another example, consider the problem of predicting human gait kinematics from motion data [46, 83]. Such systems are useful to predict gait abnormalities associated with physical or neurological deficiencies (such as amputation or stroke). The output prediction variables in such an application include right/left knee flexion, right/left ankle flexion and other displacement and rotation parameters associated with motion [83], which entails multivariate regression analysis. Further, each individual has his/her unique gait pattern [70, 80], which necessitates domain adaptation to develop user-specific prediction models, by transferring relevant knowledge from the data of other users. We intend to enhance our *DeepDAR* framework to predict multiple output variables and study its performance on these applications.

References

1. Aldecoa, R., Marín, I.: Deciphering network community structure by surprise. PLoS ONE **6**(9), e24195 (2011)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN. arXiv:1701.07875 (2017)
3. Aytar, Y., Zisserman, A.: Tabula rasa: model transfer for object category detection. In: IEEE International Conference on Computer Vision (ICCV), pp. 2252–2259 (2011)
4. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: a deep convolutional encoderdecoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **39**(12), 2481–2495 (2017)
5. Bay, H., Tuytelaars, T., VanGool, L.: Surf: speeded up robust features. In: European Conference on Computer Vision (ECCV), pp. 404–417 (2006)
6. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.: A theory of learning from different domains. Mach. Learn. **79**(1–2), 151–175 (2010)
7. Bhattarai, B., Sharma, G., Lechervy, A., Jurie, F.: A joint learning approach for cross domain age estimation. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 1901–1905 (2016)
8. Borghi, G., Fabbri, M., Vezzani, R., Calderara, S., Cucchiara, R.: Face-from-depth for head pose estimation on depth images. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2018)
9. Borgwardt, K., Gretton, A., Rasch, M., Kriegel, H., Scholkopf, B., Smola, A.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics **22**(14), e49–e57 (2006)
10. Bruzzone, L., Marconcini, M.: Domain adaptation problems: a dasvm classification technique and a circular validation strategy. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **32**(5), 770–787 (2010)
11. Cao, B., Pan, S., Zhang, Y., Yeung, D., Yang, Q.: Adaptive transfer learning. In: Association for the Advancement of Artificial Intelligence (AAAI), pp. 407–412 (2010)
12. Chapelle, O., Scholkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press (2006)

13. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. In: British Machine Vision Conference (BMVC) (2014)
14. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **40**(4), 834–848 (2018)
15. Cortes, C., Mohri, M.: Domain adaptation in regression. In: International Conference on Algorithmic Learning Theory, pp. 308–323 (2011)
16. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 886–893 (2005)
17. Dong, X., Thanou, D., Frossard, P., Vandergheynst, P.: Learning laplacian matrix in smooth graph signal representations. IEEE Trans. Signal Process. **64**(23), 6160–6173 (2016)
18. Duan, L., Tsang, I., Xu, D., Maybank, S.: Domain transfer SVM for video concept detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1375–1381 (2009)
19. Escalera, S., Fabian, J., et al., P.P.: Chalearn looking at people 2015: apparent age and cultural event recognition datasets and results. In: IEEE International Conference on Computer Vision Workshop (ICCVW), pp. 243–251 (2015)
20. Fanelli, G., Dantone, M., GallA, J., Fossati, A., Gool, L.V.: Random forests for real time 3D face analysis. Int. J. Comput. Vis. (IJCV) **101**(3), 437–458 (2013)
21. Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T.: Unsupervised visual domain adaptation using subspace alignment. In: IEEE Inrternational Conference on Computer Vision (ICCV), pp. 2960–2967 (2013)
22. Fu, Y., Huang, T.: Graph embedded analysis for head pose estimation. In: International Conference on Automatic Face and Gesture Recognition (2006)
23. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. (JMLR) **17**(1), 2096–2130 (2016)
24. Garcke, J., Vanck, T.: Importance weighted inductive transfer learning for regression. In: European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 466–481 (2014)
25. Geng, X., Zhou, Z., Miles, K.: Automatic age estimation based on facial aging patterns. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **29**(12), 2234–2240 (2007)
26. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances of Neural Information Processing Systems (NIPS), pp. 2672–2680 (2014)
27. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: an unsupervised approach. In: IEEE International Conference on Computer Vision (ICCV), pp. 999–1006 (2011)
28. Guan, Z., Li, A., Zhu, T.: Local regression transfer learning with applications to users' psychological characteristics prediction. Brain Inf. **2**(3), 145–153 (2015)
29. Guo, G., Fu, Y., Dyer, C., Huang, T.: Image-based human age estimation by manifold learning and locally adjusted robust regression. IEEE Trans. Image Process. **17**(7), 1178–1188 (2008)
30. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
31. Hu, L., Kan, M., Shan, S., Chen, X.: Duplex generative adversarial network for unsupervised domain adaptation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1498–1507 (2018)
32. Jamal, A., Namboodiri, V., Deodhare, D., Venkatesh, K.: Deep domain adaptation in action space. In: British Machine Vision Conference (BMVC) (2018)
33. Khosla, A., Sarma, A., Hamid, R.: What makes an image popular? In: International World Wide Web Conference (WWW), pp. 867–876 (2014)
34. Kingma, D., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
35. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Neural Information Processing Systems (NIPS) (2012)

36. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: Association for the Advancement of Artificial Intelligence (AAAI), pp. 2267–2273 (2015)
37. Le, T., Nguyen, K., Ho, N., Bui, H., Phung, D.: Theoretical perspective of deep domain adaptation. arXiv:1811.06199 (2019)
38. LeCun, Y., Bottou, L., Orr, G., Muller, K.: Efficient backprop. In: Neural Networks: Tricks of the Trade, pp. 9–50 (1998)
39. Liang, Z., Zhang, G., Huang, J., Hu, Q.: Deep learning for healthcare decision making with EMRs. In: IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 556–559 (2014)
40. Liu, M., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: Advances of Neural Information Processing Systems (NIPS), pp 700–708 (2017)
41. Liu, M., Tuzel, O.: Coupled generative adversarial networks. In: Advances of Neural Information Processing Systems (NIPS), pp. 469–477 (2016)
42. Liu, S., Liu, S., Cai, W., Pujol, S., Kikinis, R., Feng, D.: Early diagnosis of Alzheimer's disease with deep learning. In: IEEE International Symposium on Biomedical Imaging (ISBI), pp. 1015–1018 (2014)
43. Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: International Conference on Machine Learning (ICML), pp. 97–105 (2015)
44. Long, M., Zhu, H., Wang, J., Jordan, M.: Unsupervised domain adaptation with residual transfer networks. In: Advances of Neural Information Processing Systems (NIPS), pp. 136–144 (2016)
45. Luu, T., Low, K., Qu, X., Lim, H., Hoon, K.: An individual-specific gait pattern prediction model based on generalized regression neural networks. Gait and Posture **39**(1), 443–448 (2014)
46. Motiian, S., Jones, Q., Iranmanesh, S., Doretto, G.: Few-shot adversarial domain adaptation. In: Advances of Neural Information Processing Systems (NIPS) (2017)
47. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. International Journal of Computer Vision (IJCV) **42**(3), 145–175 (2001)
48. Pan, S., Tsang, I., Kwok, J., Yang, Q.: Domain adaptation via transfer component analysis. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 1187–1192 (2009)
49. Pan, S., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. (TKDE) **22**(10), 1345–1359 (2010)
50. Pan, S., Tsang, I., Kwok, J., Yang, Q.: Domain adaptation via transfer component analysis. IEEE Trans. Neural Netw. Learn. Syst. (TNNLS) **22**(2), 199–210 (2011)
51. Pardoe, D., Stone, P.: Boosting for regression transfer. In: International Conference on Machine Learning (ICML), pp. 863–870 (2010)
52. Peng, K., Wu, Z., Ernst, J.: Zero-shot deep domain adaptation. In: European Conference on Computer Vision (ECCV), pp. 793–810 (2018)
53. Pinto, H., Almeida, J., Goncalves, M.: Using early view patterns to predict the popularity of youtube videos. In: ACM International Conference on Web Search and Data Mining (WSDM), pp. 365–374 (2013)
54. Puheim, M., Madarasz, L.: Normalization of inputs and outputs of neural network based robotic arm controller in role of inverse kinematic model. In: IEEE International Symposium on Applied Machine Intelligence and Informatics, pp. 35–38 (2014)
55. Qi, F., Yang, X., Xu, C.: A unified framework for multimodal domain adaptation. In: ACM Multimedia Conference (ACM-MM), pp. 429–437 (2018)
56. Ranganathan, H., Venkateswara, H., Chakraborty, S., Panchanathan, S.: Deep active learning for image classification. In: IEEE International Conference on Image Processing (ICIP), pp. 3934–3938 (2017)
57. Ranganathan, H., Venkateswara, H., Chakraborty, S., Panchanathan, S.: Multi-label deep active learning with label correlation. In: IEEE International Conference on Image Processing (ICIP), pp. 3418–3422 (2018)
58. Rothe, R., Timofte, R., VanGool, L.: Deep expectation of real and apparent age from a single image without facial landmarks. Int. J. Comput. Vis. (IJCV) **126**(2–4), 144–157 (2018)

59. Rozantsev, A., Salzmann, M., Fua, P.: Residual parameter transfer for deep domain adaptation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4339–4348 (2018)
60. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: European Conference on Computer Vision (ECCV), pp. 213–226 (2010)
61. Sankaranarayanan, S., Balaji, Y., Castillo, C., Chellappa, R.: Generate to adapt: aligning domains using generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8503–8512 (2018)
62. Sener, O., Song, H., Saxena, A., Savarese, S.: Learning transferrable representations for unsupervised domain adaptation. In: Advances of Neural Information Processing Systems (NIPS), pp. 2118–2126 (2016)
63. Shen, J., Qu, Y., Zhang, W., Yu, Y.: Wasserstein distance guided representation learning for domain adaptation. In: Association for the Advancement of Artificial Intelligence (AAAI), pp. 4058–4065 (2018)
64. Sherrah, J., Gong, S.: Fusion of perceptual cues for robust tracking of head pose and position. Pattern Recogn. **34**(8), 1565–1572 (2001)
65. Smola, A., Kondor, R.: Kernels and regularization on graphs. In: Conference on Computational Learning Theory (COLT), pp. 144–158 (2003)
66. Sriperumbudur, B., Fukumizu, K., Lanckriet, G.: Universality, characteristic kernels and RKHS embedding of measures. J. Mach. Learn. Res. (JMLR) **12**, 2389–2410 (2011)
67. Sriperumbudur, B., Gretton, A., Fukumizu, K., Scholkopf, B., Lanckriet, G.: Hilbert space embeddings and metrics on probability measures. J. Mach. Learn. Res. (JMLR) **11**, 1517–1561 (2010)
68. Steinwart, I.: On the influence of the kernel on the consistency of support vector machines. J. Mach. Learn. Res. (JMLR) **2**, 67–93 (2001)
69. Stevenage, S., Nixon, M., Vince, K.: Visual analysis of gait as a cue to identity. Appl. Cognit. Psychol. **13**(6), 513–526 (1999)
70. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: Association for the Advancement of Artificial Intelligence (AAAI), pp. 2058–2065 (2016)
71. Szabo, G., Huberman, B.: Predicting the popularity of online content. Commun. ACM **53**(8), 80–88 (2010)
72. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
73. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Simultaneous deep transfer across domains and tasks. In: IEEE International Conference on Computer Vision (ICCV), pp. 4068–4076 (2015)
74. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7167–7176 (2017)
75. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: maximizing for domain invariance. arXiv preprint arXiv:1412.3474 (2014)
76. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5018–5027 (2017)
77. Venkateswara, H., Lade, P., Ye, J., Panchanathan, S.: Coupled support vector machines for supervised domain adaptation. In: ACM Multimedia Conference (ACM-MM), pp. 1295–1298 (2015)
78. Volpi, R., Morerio, P., Savarese, S., Murino, V.: Adversarial feature augmentation for unsupervised domain adaptation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5495–5504 (2018)
79. Winter, D.: The Biomechanics and Motor Control of Human Gait: Normal, Elderly, and Pathological. University of Waterloo Press (1991)
80. How to improve neural network stability and modeling performance with data scaling. https://machinelearningmastery.com/how-to-improve-neural-network-stabilityand-modeling-performance-with-data-scaling/

81. Yamada, M., Sigal, L., Chang, Y.: Domain adaptation for structured regression. Int. J. Comput. Vision **109**(1–2), 126–145 (2014)
82. Yan, K., Zheng, W., Cui, Z., Zong, Y., Zhang, T., Tang, C.: Unsupervised facial expression recognition using domain adaptation based dictionary learning approach. Neurocomputing **319**, 84–91 (2018)
83. Yun, Y., Kim, H., Shin, S., Lee, J., Deshpande, A., Kim, C.: Statistical method for prediction of gait kinematics with gaussian process regression. J. Biomech. **47**(1), 186–192 (2014)
84. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances of Neural Information Processing Systems (NIPS), pp. 649–657 (2015)
85. Zhou, D., Scholkopf, B.: A regularization framework for learning from graph data. In: International Conference on Machine Learning Workshop (2004)
86. Zhu, R., Sang, G., Zhao, Q.: Discriminative feature adaptation for cross-domain facial expression recognition. In: International Conference on Biometrics (ICB) (2016)

Deep Learning-Based Pedestrian Detection for Automated Driving: Achievements and Future Challenges

Michelle Karg and Christian Scharfenberger

Abstract Deep learning is considered as a key technology for the development of advanced driver assistance systems and future automated driving. Focus lies especially on the perception of the environment by camera, Radar, and Lidar sensors and fusion concepts. Camera-based perception includes the detection of road users. Highest detection performance is especially required for detecting vulnerable road users such as pedestrians and bicycle drivers. Here, tremendous improvement in vision-based object detection has been achieved within the past decade. Research on object detection has been stimulated by public datasets. The results on public benchmarks show the progress of pedestrian detectors from hand-crafted features, over part-based models towards deep learning. The gap between human and machine performance becomes smaller, leading to the question whether pedestrian detection is solved when the detection performance reaches human performance? As false detections can lead to hazardous situations in traffic scenarios, the expectations on the performance of artificial intelligence for advanced driver assistance systems and automated driving often go beyond human performance. Challenges are precise localization, occlusion, distant objects, and corner cases, where only little or no training data is available. To foster research in this direction, a new comprehensive dataset for pedestrian detection at night has been released. This chapter first introduces vision-based perception of road users as a safety-critical application with increasing demand on detection performance. In the second part, it summarizes concepts for pedestrian detection, including an overview on public datasets and evaluation metrics. The dependency between task complexity and task performance is discussed. Based on this discussion, challenges in pedestrian detection are identified and future directions are outlined. Further improvements in performance can be achieved by including components such as tracking, scene understanding, and sensor fusion. In conclusion, the application of deep learning to advanced driver assistance systems and automated driving is driven

M. Karg (✉) · C. Scharfenberger
Continental AG, Lindau, Germany
e-mail: michelle.karg@continental-corporation.com

C. Scharfenberger
e-mail: christian.scharfenberber@continental-corporation.com

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_5

by the goal of achieving safe maneuvering in any traffic scene, any weather condition, and under real-time constraints. This places high demands on the development of deep network architectures.

Keywords CNN · Pedestrian detection · Advanced driver assistance systems · Automated driving

1 Introduction

Deep learning has become a key technology in computer vision since its introduction and has found many applications [1–7]. The development of deep learning can be followed by the early works on Restricted Boltzmann Machines (RBM), Deep Belief Networks (DBN), and supervised pre-training, over more recent works on deep Convolutional Neural Networks (CNNs) for computer vision, and towards future directions such as Differentiable Neural Computers (DNCs) [2, 3, 5, 7–16]. Deep learning is based on the concept of neural networks and extends the design of traditional neural networks by representation learning, stochastic gradient descent, deep architecture, computationally efficient nonlinearities, and the use of large datasets and fast computing. Representation learning optimizes the feature representation needed for classification by end-to-end learning; the feature representation and the classification are learned by a common training without the need for hand-designed features [2]. Increasing the depth of neural networks helps to model nonlinearities, where deep networks can model complex functions with less parameters than shallow and broad networks [11, 17]. Stochastic gradient descent (SGD) enables training of deep network architectures, where in each mini-batch the gradient is estimated and the weights are updated. SGD supports the use of large training datasets, because the dataset is split into mini-batches and gradients can be computed for the mini-batches only. In combination with fast processing, e.g. use of GPUs, deep networks can be trained on large datasets such as ImageNet leading to superior generalization performance than achieved with many other techniques based on hand-designed features and domain expertise [2, 5, 7, 18]. In many cases, rectified linear units or related units are used as activation function achieving faster convergence of the training for deep networks than the use of, e.g., the sigmoid function used for shallow networks in the past [19]. In addition to the changes in the network design, the amount of data used for training, validating, and testing deep networks has been increased, e.g., reaching over a million samples in the ImageNet dataset [17]. Similarly, the amount of computing resources required to run such highly-efficient deep networks has increased, with the support of parallel computing playing a key role, e.g., by GPUs.

Deep learning helped to improve the detection and classification performance in many domains in computer vision, and solved issues in areas where approaches based on hand-crafted features provided only limited solutions [2, 5, 7, 18]. One such important area is the development of advanced driver assistance systems and automated driving with steadily increasing requirements on the detection performance

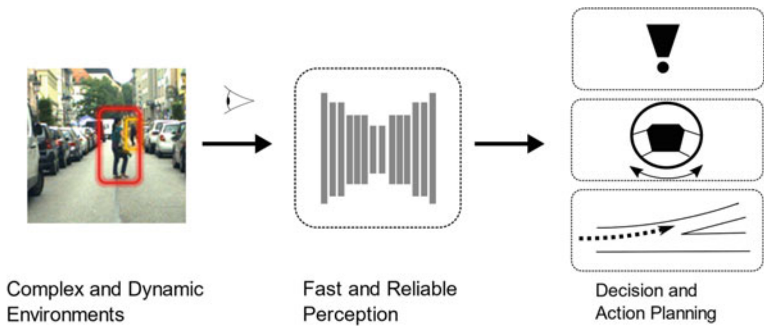

Complex and Dynamic Environments
Fast and Reliable Perception
Decision and Action Planning

Fig. 1 Highly accurate sensor systems are important for developing automotive systems for warning the driver, planning emergency steering, or planning driving strategies

of sensor systems in complex and challenging scenarios. Focus lies especially on the perception of the environment by cameras, Radar, or Lidar sensors. These sensors have to—reliably and fast—extract the environment information in any scenario to allow the vehicle to warn drivers against critical situations, to perform emergency steering for preventing accidents autonomously, and to plan driving paths in the area of automated driving as illustrated in Fig. 1. Given the high requirements towards the detection performance of today's and tomorrow's sensor systems with focus on computer vision and camera sensors, deep learning is considered as essential for advanced driver assistance systems and automated driving.

Camera-based sensing for advanced driver assistance systems and automated driving includes the detection of road users. Highest detection performance is especially required for detecting vulnerable road users such as pedestrians and bicycle drivers. Here, tremendous improvement in vision-based object detection has been achieved within the past ten years. Research on object detection has been stimulated by many public datasets, such as the Caltech dataset [20, 21]. Benchmarks show the progress of pedestrian detectors starting from hand-crafted features, over part-based models towards deep learning. Recent benchmarks and comparisons of CNN-based detection approaches against human detection performance show a fast closing gap between human and machine performance [5, 22–24]. This trend may raise the question whether the detection of pedestrians and bicyclists can be considered as sufficiently solved when the detection performance reaches human performance?

As both false positive and false negative detections can lead to hazardous situations in traffic scenarios, the expectations on the performance of artificial intelligence for advanced driver assistance systems and automated driving often go beyond human performance. The expectations are even higher when including precise localization of objects, occlusion, very small and distant objects, sudden appearance of pedestrians such as playing children running onto the road surface and corner cases, where only little or no training data is available [25–27]. Here, artificial intelligence for advanced driver assistance systems can help the driver in critical situations by monitoring the environment.

This chapter aims to introduce vision-based detection of road users as a safety-critical application. Here, the demand on the detection performance increases for an increasing level of automatization. Therefore, the different automatization levels for advanced driver assistance systems and autonomous driving are summarized. This leads to the question of what has been achieved in the field of pedestrian detection and what are future challenges to further improve the detection performance. As pedestrian detection is closely related to vision-based object detection, deep learning-based approaches for solving this task are summarized. Performance depends both on the algorithmic solution and the task complexity. For this reason, the relationship between task complexity and error rate is discussed, and factors that influence task complexity are described. Considering pedestrian detection, several public datasets have been released in the past. These datasets differ in task complexity, e.g., in size of the dataset, variability of the environment (number of cities, countries), day or night data, and granularity of annotation. Afterwards, evaluation metrics for pedestrian detection are presented. The performance for pedestrian detection for commonly used datasets are summarized and challenges for pedestrian detections are outlined. While this chapter focuses on CNN-based single-frame pedestrian detection, further improvement in detection performance can be achieved by including tracking, scene understanding, and sensor fusion.

In a nutshell, the application of deep learning approaches to advanced driver assistance systems and automated driving is driven by the goal of safe and reliable maneuvering in any complex traffic scene and any weather condition under real-time constraints, placing high demands on the development of deep network architectures.

2 Safety Relevance of Pedestrian Detection for Advanced Driver Assistance Systems and Automated Driving

The discussion about passengers' and road users' safety and driving comfort started almost right after the invention of the automobile by Carl Benz in 1886. The focus on addressing safety aspects was merely on the passengers' safety inside of a vehicle in the past decades, and initiated many technical improvements and advances increasing the road safety and driving comfort. Prominent examples include anti-lock systems and airbags, greatly reducing fatalities by ensuring the steerability of vehicles in critical situations or preventing serious injuries during accidents.

The early advantages in radar and camera-based driver assistance systems had a positive impact on developing novel advanced comfort and safety functions and improved road safety further. Many of these driver assistance functions had the task to warn drivers on potentially dangerous traffic situations and to prevent collisions with other road users such as vehicles while manually steering the car. The technologies developed put a great emphasis on passengers' safety within a vehicle and helped reduce the number of fatalities significantly despite the increasing number of vehicles on public roads.

The upcoming new wave of advanced driver assistance systems and systems supporting automated driving together with new regulations and legislations shifted the focus of road safety into a new direction. While there was a strong focus on ensuring passengers' safety within the vehicle, emphasis is now put on better protecting vulnerable road users such as pedestrians, bicyclists and motorbikes outside of the vehicle. The introduction of different levels of automated driving by the National Highway Traffic Safety Administration—the SAE levels 0–5—intensified this trend further by moving the responsibility of safe driving and protecting vulnerable road users in any situation from the driver to the vehicle with an increasing SAE level. The following overview summarizes briefly the different levels of automated driving:

– Level 0: The human driver controls and steers the entire vehicle. Assistance systems may be available to warn drivers against critical situations.
– Level 1: The driver controls most of the vehicle functions, but the vehicle performs specific functionality like automatically accelerating, deceleration, or steering.
– Level 2: Level 2 considers driver assistance systems that perform automatic steering, acceleration and deceleration using environment information. The driver can have his hands off the steering wheel and foot off the pedal a certain time but must be ready to take over control anytime.
– Level 3: Level 3 extends Level 2 by shifting safety-critical functions from the driver to the vehicle under certain environment and traffic conditions. The driver still has to be present and intervene but does not have to monitor the situation in the same way as for previous levels.
– Level 4: Level 4 vehicles are meant to be largely automated. They can perform all safety-critical driving functions and monitor all roadway conditions for an entire trip. While the level of automation is high, it is still limited to the operational design domain and does not cover each and every driving scenario.
– Level 5: This level expects the vehicle's system performance to equal that of a human driver in any environment and driving scenario, with no driver interventions needed at all.

All levels of automation impose strong requirements towards the detection of vulnerable road users. Here, reliable pedestrian detection became a crucial task for advanced driver assistance systems and automated driving, starting with Level 0, where safety functions can warn drivers in critical situations, and up to Level 5, where no more driver control is needed in the vehicle.

Besides the increasing level of automation, the need for highly reliable pedestrian detection is also visible in traffic accident reports, where many injuries to persons in road traffic occur in collisions between vehicles and pedestrians. Here, the accident report of 2017 [28] of the Federal Statistical Office of Germany is selected and the related statistics are summarized.

More than two million accidents were reported in Germany in 2017, with injuries to persons in 302,656 accidents [28]. Most injuries to persons happened in urban areas as shown in Fig. 2, whereas the number of injuries on motorways is smaller but with a slight increase in severity. Pedestrians and bicycle drivers were involved in 111,715 cases, approximately one third of the cases. As such, reliable pedestrian

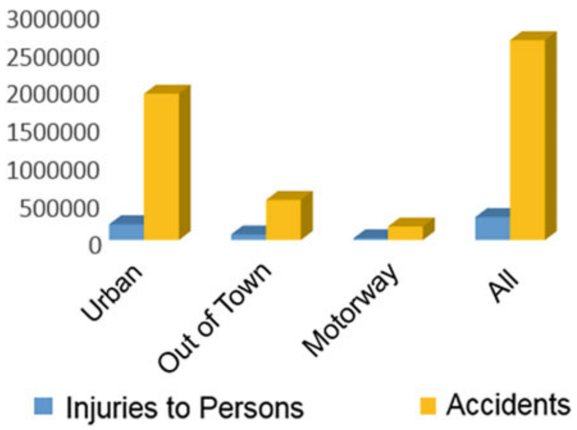


Fig. 2 Accident report from 2017 of the Federal Statistical Office, Germany [28]

detection is of great importance to reduce the number of vehicle-pedestrian collisions in urban traffic.

Reliable pedestrian detection is considered as a challenging task in computer vision and machine learning. Figure 3 shows a typical road scenario in an urban environment and exhibits several challenges to pedestrian detection. Examples include pedestrians in a crowd, pedestrians and children hidden behind parked vehicles or covered by other pedestrians or items. Given the scenario outlined in Fig. 3, the most important requirements towards pedestrian detection for assistance and self-driving systems can be summarized as follows:

1. Reliable pedestrian detection must work at any lighting condition, including day- and night-time, sunset, dawn and shallow sun.
2. Reliable pedestrian detection must work in the far and near surrounding of a vehicle.
3. Pedestrian detection is robust against different sizes, poses, appearances and views.

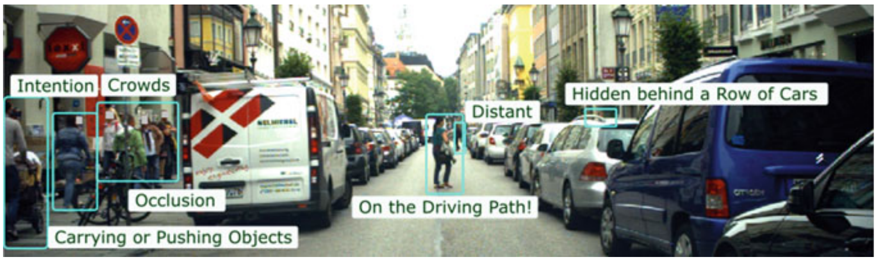


Fig. 3 Typical road scenario in an urban environment

4. Robust detection can address challenging environment and weather conditions, including light and heavy rain, snow, fog, sun, and a high dynamic in the ambient illumination such cast shadows.
5. The detection works reliably in complex environment and traffic situations.
6. Pedestrians can be detected robustly even when they are covered by carried objects, vehicles or other persons.
7. An approach to pedestrian detection can detect individual pedestrians in a crowd and extract the most relevant and critical pedestrian that a vehicle may need to brake for.

The basis for reliable detection is the extraction of environment and traffic-relevant information from sensor data. To meet the requirements for robustly pedestrian detection, systems supporting assisted and automated driving combine and fuse input from several sensors. The most common setups process either individual or fused data from Lidar, Radar, and camera sensors. Focusing on camera sensors, the first step includes the detection of objects in single images. Object detection may include pedestrians, vehicles, traffic lights and traffic signs among others. Since single frame detection may be very sensitive to false positive or false negative being critical for safety-relevant functions, post-processing such as tracking is often required to increase the robustness of object detection by leveraging spatio-temporal information or input from data sources such as Radar or Lidar for plausibility checks. Sensor data fusion and post-processing can increase the overall detection accuracy by improving the detection confidence through temporal, spatial and multimodal coherence, and by correcting false detections through time series analysis. However, a high detection accuracy of a single sensor in its working range is crucial to make sensor fusion highly efficient. As such, this article focuses on the examination of the overall performance of a single-sensor, single-frame approach for detecting pedestrians based on convolutional neural networks (CNN).

3 Pedestrian Detection

Object detection includes both the classification of an object and its localization in an image. Pedestrian detection can be considered as a special case of object detection, where the classification reduces to a binary decision: ‘pedestrian’ or ‘background’. The localization task distinguishes object detection from object classification. The latter is based on image crops each containing an object, the former is based on images that may include objects or background only. As object detection requires searching for potential object candidates/object proposals in a complete image, it is computationally more expensive than object classification and more prone to the detection of false positives. The separation in classification and object detection is visible in the design of CNN architectures for both tasks. CNNs for object classification serve as core networks in the larger CNN architectures for object detection.

For this reason, this chapter starts with summarizing two common approaches for CNN architectures for object detection. Then, a review follows on core networks for feature extraction and classification. Since the detection performance of object detectors highly relates to task complexity, factors influencing task complexity are presented and discussed additionally. Afterwards, an overview of publicly available datasets for research on pedestrian detection is provided, and evaluation metrics for pedestrian detection are summarized. This chapter concludes with a discussion on the performance of CNNs, open challenges for pedestrian detection, and future directions.

3.1 Convolutional Neural Networks for Pedestrian Detection

The task of detecting objects can be broken down into several sub-tasks, including object localization, object classification, and estimating a detection confidence. These sub-tasks can be described as follows:

1. Localization of an object l(obj) = f(x, y, width, height): Where is the object located in an image, and what is its size?
2. Classification of the object c(obj) = h(cj, x) given the observation x and c classes: Which class is assigned to the object?
3. Estimation of the confidence z(obj) = P(cj | x): How confident is the detector in its detection?

Pedestrian detection has been addressed in computer vision by different approaches over time as illustrated in Fig. 4. As direct classification of pixel values is easily prone to errors, features based on edges or structures were introduced. Histogram normalization increased the robustness of these features against noise,

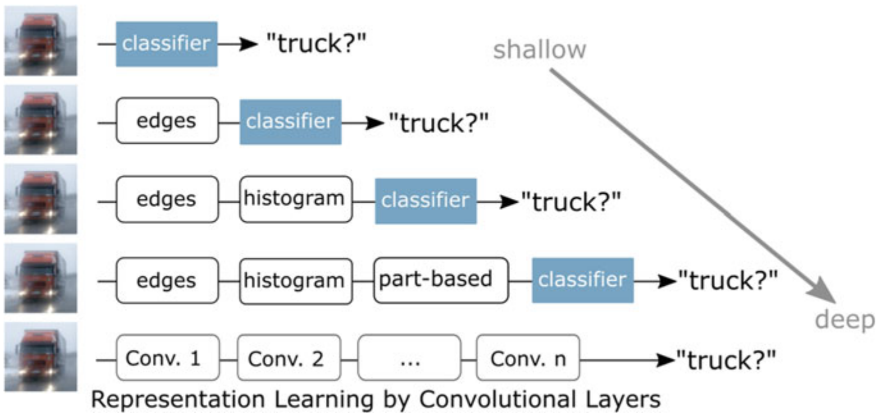


Fig. 4 Object detection: from shallow architectures towards deep architectures learned end-to-end

lighting, and spatial transformations [12]. The development of part-based models improved the detection of objects further, considering difficult cases such as occlusions, difficult poses, or rare viewing angles [29]. All approaches to pedestrian detection relied on well-crafted, hand-designed features, and the performance of pedestrian detection and classification depended strongly on an appropriate set of features chosen. An overview on hand-designed features especially suited for pedestrian detection is provided in the work of Dollar et al. [21].

End-to-end learning of deep neural networks improved pedestrian detection further, where a training approach based on stochastic gradient descent learns the feature representation and the classification jointly [26, 30]. This is also referred to as representation learning. Representation learning based on convolutional neural networks (CNN) achieves higher performance and better generalization than designing hand-crafted features for many object detection tasks and is, hence, applied to many detection tasks in biomedical, robotic, and automotive applications.

Convolutional neural networks for object detection can be subdivided into two main approaches (as shown in Fig. 5):

– Single-stage architectures, where multi-class classification and localization is applied directly after feature extraction.
– Two-stage architectures consist of two network heads sharing the encoder: (1) a region proposal network including binary classification and localization, and (2) a classification network for multi-class classification.

Both architectures are based on an encoder, which consists of a set of convolutional layers that computes the feature representation of an image and is learned end-to-end. The feature representation includes low-level features such as gradients and corners, mid-level features representing simple shapes such as lines, curves, and circles, and high-level features representing object parts. Computing the feature representation

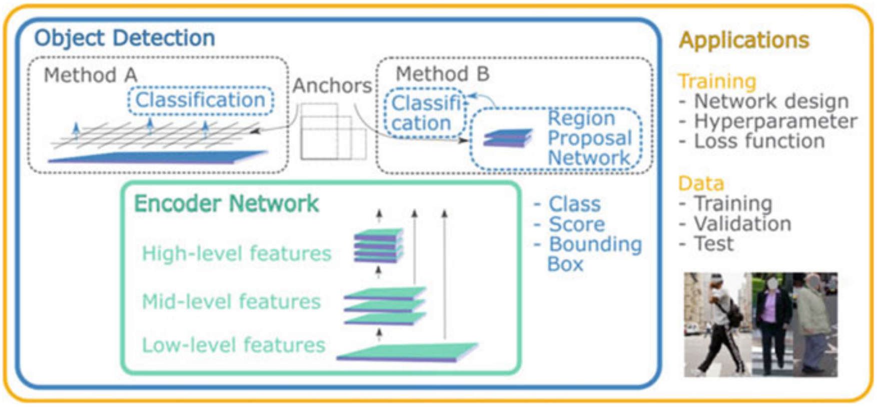


Fig. 5 Object detection: single-stage and two-stages approaches

once for an entire image saves run-time [31]. Furthermore, the single and the two-stage architectures have a set of predefined anchors in common specifying priors for the aspect ratio and scale of the objects.

Single-stage architectures include architectures such as Yolo [32, 33] and SSD [34]. After the convolutional layers for feature representation follow additional convolutional layers for multi-class classification and bounding box regression. Yolo and SSD differ in their use of low-level features for classification and in their loss functions. Yolo includes low-level features using a re-organization layer, while SSD applies classification and bounding box regression to several feature resolutions. Furthermore, the prediction of bounding boxes is constrained in Yolo using logistic activations. The loss function of SSD is a weighted sum of the localization and the confidence loss. Yolo additionally outputs separate confidence scores for each prediction. The confidence scores are trained by using an additional loss component that relates to the confidence prediction with the intersection-over-union (IoU) between prediction and ground truth.

Two-stage architectures include architectures such as Fast R-CNN [31], Faster R-CNN [3] and R-FCN [35]. These architectures include two neural network heads after the encoder network: a region proposal network (RPN) and a classification network.

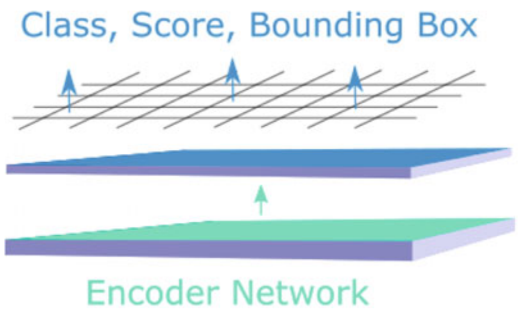
The region proposal head consists of a binary classification (object vs. no object) and bounding box regression. The RPN localizes potential object proposals (or also called object candidates) and forwards proposals with a sufficiently high score to the classification network. Non-maxima suppression (NMS) can be applied to reduce the number of possible object proposals. Furthermore, a region-of-interest pooling layer is required to transform each crop of an object proposal to a fixed size [3, 31, 35].

The classification network can consist of a set of fully connected layers for the Faster R-CNN [3, 31] or convolutional layers trained end-to-end for the R-FCN [35].

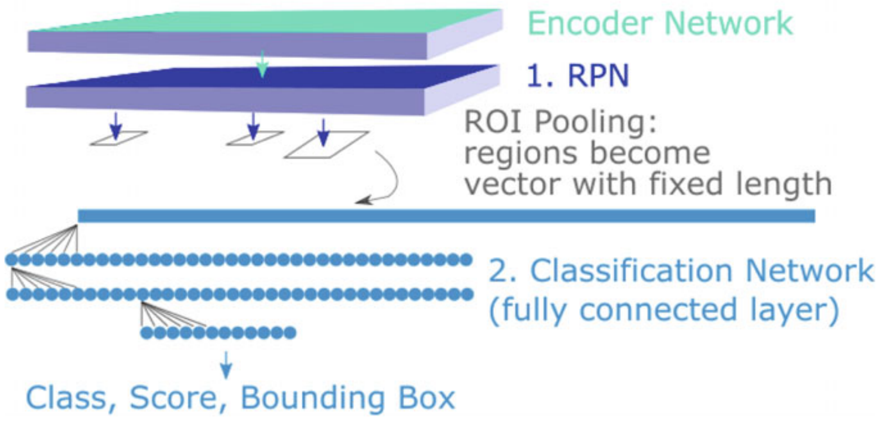
Following [4], single stage architectures have the advantage of faster computation, whereas two-stage architectures can achieve higher accuracies. The differences in the network design of the two architectures are visualized in Fig. 6a, b.

The detection performance depends on the chosen encoder for feature representation. Several basic network designs have been proposed as encoders:

- The **AlexNet** has been introduced in 2012 and consists of five convolutional layers and three fully connected layers [5]. The number of channels in each layer is 48-128-192-192-128 for the convolutional layers, and 2048-2048-1000 for the fully connected layers. It has 60 million parameters and, to avoid overfitting, data augmentation and drop-out are applied during training. The AlexNet achieved an error rate lower than conventional classification approaches based on hand-designed features on the ImageNet LSVRC-2010 contest.
- The **VGGnet** introduces the use of only small filter sizes, i.e., 3×3, for CNN-based classification [6]. Network architectures of different depth ranging between 16 and 29 layers are compared in [6]. The number of channels is doubled after



(a) Single stage architecture


Encoder Network
1. RPN
ROI Pooling: regions become vector with fixed length
2. Classification Network (fully connected layer)
Class, Score, Bounding Box

(b) RPN and classification network

Fig. 6 Single stage architecture and two-stage architecture for object detection. The RPN of the two-stage architecture is similar to the single-stage architecture; yet, the RPN classifies only between background and object, whereas the output of the single-stage architecture is a multi-class classification

max-pooling starting with 64 channels in the first layer and ending with 512 channels in the last layer. The three fully connected layers contain 4096-4096-1000 parameters. The number of parameters range between 133 million for the small VGG-A network with 11 convolutional layers to 144 million for the large VGG-E network with 19 convolutional layers. The additional depth of the VGG-22 (VGG-E) network leads to 4.1% less top-1 error rate on the LSVRC dataset in comparison to the VGG-14 (VGG-A) network.

– **ResNets** are based on stacking residual layers and are fully convolutional [7]. Residual layers facilitate the learning of deep networks by addressing the degradation problem: the learned function is modeled as F(x) + x, with F(x) the residual and x the identity. The residual F(x) is learned in the ResNet. This residual

mapping is represented by branched sub-network architectures. One branch is a shortcut connection skipping layers in the sub-network to represent the identity x. The other branch is a set of layers for learning the residual F(x). Common ResNet architectures include ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152 [7]. In [36], up to 1000 layers are trained using a residual architecture and stochastic depth during training to avoid vanishing gradients.
– Inception modules are introduced with the **GoogleNet/InceptionNet-v1** [13]. An inception module consists of parallel branches with different filter size, e.g., 1×1, 3×3 and 5×5. 1×1 filters are added to the network design for dimensionality reduction. InceptionNet-v1 and **InceptionNet-v3** are 22 layers and 42 layers deep, respectively. InceptionNet-v3 comes with computation costs that are 2.5 times higher than the costs of InceptionNet-v1 [13, 14]. InceptionNet-v3 further replaces 5×5 filters by two stacked 3×3 convolutions to save runtime [14]. These filters can be factorized into asymmetric convolutions ($1 \times n$, $n \times 1$), and expanded filter bank outputs can be added to late layers. Training of inception networks can be accelerated by auxiliary classifiers or residual connections [14, 15]. The combination of the two concepts for residual and inception layers is investigated in [15].

Detection approaches based on convolutional neural networks have the potential to improve the performance of pedestrian detection in automotive significantly. Applying detection in the automotive domain comes with many constraints and requirements towards functionality and safety. These constrains impose important design criteria to consider when designing convolutional neural networks for automotive applications:

– Fast inference of the network is required to support integration in embedded systems.
– Sensor systems for automotive applications require multi-class detection for perception of the complete traffic scene. This includes other object classes such vehicles, traffic signs, road construction objects, and road boundaries besides pedestrians, bicyclists, and motorcyclists.
– Highly accurate localization of the objects and precise estimates of the object size are required for traffic scene understanding and automated decision making.
– Meaningful confidence values for detections are relevant for safety-critical decision making in systems supporting advanced driver assistance and automated driving.
– Furthermore, high precision and high robustness in terms of few false positives and false negatives is required for safety-critical decision making.

Beyond those design criteria, such pedestrian detectors will run in highly complex and dynamic environments and face the detection of safety-critical situations. This raises additional demands and challenges on securing the performance of such detectors learned end-to-end [37].

3.2 *The Dependence of the Error Rate on Task Complexity*

The performance of an object detector depends on task complexity. With increasing task complexity, low error rates are harder to achieve. Achieving low error rates is important for safety-relevant applications in the automotive domain. An example for the relationship between error rate and task complexity is provided by comparing classification results for different public benchmarks first, followed by a description of different aspects influencing task complexity.

Figure 7 shows an example of the relationship between error rate and task complexity by comparing the classification error for different classification benchmarks. The benchmarks considered include object classification for

- gray-scale images and digits only (MNIST dataset, 70,000 images) [38],
- street view house numbers (SVHN dataset, 600,000 images) [39]
- 10 or 100 categories including classes such as cars, airplanes, ships, animals (CIFAR-10, CIFAR-100, each 60,000 images) [40]
- 1,000 categories (ImageNet, >1,000,000 images) [17]; here, the top-5 error rate is reported besides the top-1 error rate because the classifier may correctly recognize objects in the background of an image crop. For the top-5 error rate, it is sufficient when the correct label is among the 5 highest scored predictions for an image.

Figure 7 shows the best results for CNN-based object classifiers for the different datasets. Error rates below 1% can be achieved for the MNIST dataset. With increasing task complexity, the demand on the representational power of the classifier, its ability to converge during training, and on the size and quality of the training dataset increases. Hence, achieving top-1 error rates below 1% becomes more challenging. For the ImageNet dataset, CNNs can achieve better performance than humans, especially for classes such as breeds of dogs where human ratings depend on the familiarity with dog breeds. Best performance is achieved for ensemble network

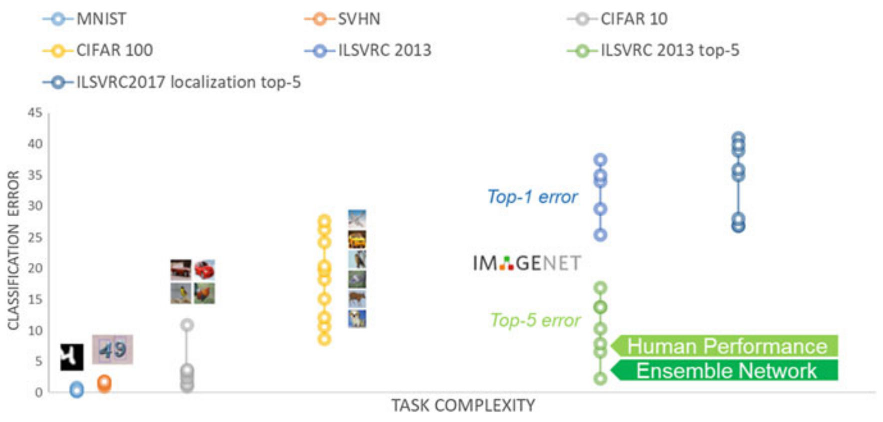


Fig. 7 Relationship between task complexity and classification error rate

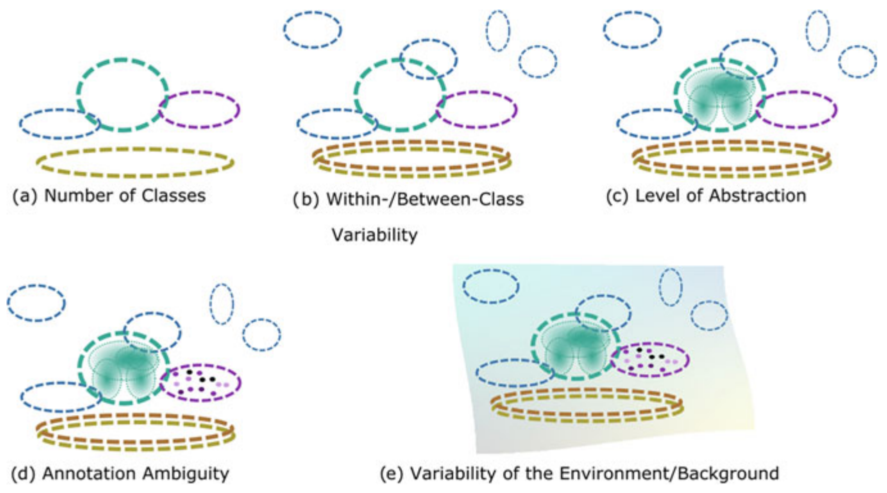

(a) Number of Classes
(b) Within-/Between-Class Variability
(c) Level of Abstraction
(d) Annotation Ambiguity
(e) Variability of the Environment/Background

Fig. 8 Visualization of task complexity in a two-dimensional feature space

architectures. Ensemble network architectures consist of several networks, and e.g., majority voting can be used as final prediction.

Furthermore, the probability of a correct classification by chance decreases with increasing number of classes. The probability of correct classification by change is 10% for 10 categories, 1% for 100 categories, and 0.1% for 1,000 categories. The kappa statistic normalizes the success rates by chance and provides a metric how much a detector is better than 'throwing a dice by chance'. The computed kappa statistic of the MNIST dataset ($\kappa = 99.4$) is higher than kappa statistic of the ImageNet dataset ($\kappa = 80.6$), indicating that the number of classes is not the only property impacting task complexity.

Other important properties influencing task complexity are summarized in the following and illustrated in Fig. 8:

– **Number of classes**: Task complexity increases with number of classes.
– **Within- and between-class variability**: Large between-class variability simplifies task complexity by forming easily separable feature clusters with strong class relationships. Large within-class variability can lead to overlapping feature clusters that can make the learning of decision hyperplanes difficult and can increase the task complexity.
– **Level of class abstraction**: The granularity of the class definition further influences task complexity. Examples related to pedestrian detection from low to high level of abstraction are: (a) bounding box representation for pedestrian, (b) subdivision into elderly, adult and child, or (c) subdivision into groups such as policemen, border officer, etc. Task complexity increases for subclass definitions where feature spaces overlap and lead to confusions.
– **Annotation ambiguity**: Task complexity also depends on annotation ambiguity. When detecting objects in images, the annotation of the objects is in most cases

clear. Exceptions can occur for occlusions or objects with low resolution. Annotating can become prone to ambiguity for other tasks such as gesture and action recognition, intention recognition, and emotion recognition. For action recognition, the on-/offset of an action may be ambiguous [34]. For gesture recognition, the interpretation of an action as a gesture may lead to ambiguity, e.g., a pedestrian in the distance holding his arms up: Is a pedestrian waving or stretching his arm? Deducing the intention from a gesture or action can become even more complicated, e.g., a pedestrian waving in the distance: Is a pedestrian waving goodbye to a friend or signaling to be recognized? When considering emotion recognition, several approaches for defining emotions exist in literature, e.g., either categories or dimensions [33]. Furthermore, emotions can blend, and more than a single emotion can be displayed. The possibility of annotation ambiguity can be ordered in the following way after increasing risk of ambiguity: annotating objects, actions, gestures, intentions, emotions.

– **Variability of the environment/background**: Introducing the background class increases task complexity by two additional factors:

  a. Complex background can lead to structures in the feature space that are similar to objects and cause false positives. Prominent examples include structure of branches and leaves of trees that are prone to false positives.
  b. The amount of background information usually predominates in natural scenes; when scanning an image for objects, the large number of possible regions for objects can lead to a risk of additional false positives in comparison to pure object classification, where only pre-clipped image crops are used.

Beyond these properties related to task complexity, additional complexity relates to the availability of training samples. Here, the feasibility of collecting, storing, and annotating large image datasets is the dominating factor.

In addition, task complexity directly relates to learning decision hyperplanes. The more difficult the task, the more complex the decision hyperplanes. When using CNNs, the depth and the number of network parameters increase for more complex decision hyperplanes. When considering the network depth, the vanishing gradient problem is a limiting factor and guides the design of the network architecture [7, 13, 14].

Pedestrian detection is considered a highly complex task because of the large variability of persons in their appearance and posture, and the complex and dynamic background in natural scenes. The high variability results in a high within-class variability and imposes challenges on training and learning decision hyperplanes. Larger pedestrian datasets aim to address this issue by providing more variability in their data, but task complexity for pedestrian detection increases further when introducing

a. a high variety of different postures and appearances, including occlusions, groups or crowds of people, and pedestrians carrying or pushing objects
b. high variety of different illuminations from data recorded at daytime, dawn, and night

c. many different background scenes by collecting data from different cities and countries.

The next chapter provides an overview of public available research datasets for pedestrian detection that address the challenges and aim to foster research in their respective domains.

3.3 *Overview on Public Research Datasets for Pedestrian Detection*

Early research datasets for pedestrian detection such as **INRIA** [12], **ETH** [41] and **TUD-Brussels** [42] provide several thousand images collected at daytime with no track identifiers to enable early research on pedestrian detection. The **Daimler** dataset is a magnitude larger and can be extended by a set of stereo image pairs [43]. The images are gray-scale. The **Caltech** pedestrian dataset was published in 2009 and has been widely used for pedestrian research since then [20, 21]. It includes 249.884 images collected in the greater Los Angeles metropolitan area at daytime. The dataset is split equally into images for training and testing, and track identifiers are provided.

Further information on the pose of pedestrians is included in the datasets Kitti [44], CityPersons [45], NightOwls [26], EuroCity [46] and nuScenes [47]. The **Kitti** dataset includes 14.999 frames and is suited for multi-class detection and sensor fusion for traffic scenes [44]. The annotations include pedestrians, cyclists and cars. Additional input can be stereo, optical flow, and laser points. Recently, the **nuScenes** dataset has been released, which is a large-scale multi-modal dataset and is inspired by the Kitti dataset [47]. It includes recordings from the sensors Radar, Lidar, and 8 cameras, and provides a 360 degree field of view. The **CityPersons** dataset includes 8.475 of highly diverse images [45]. Even though the number of images is small in comparison to larger datasets such as Caltech, NightOwls and EuroCity, high complexity of the dataset is achieved by selecting very diverse scenes, images with many pedestrians and groups, and good annotation quality. The **EuroCity** dataset has been recently published and includes 47.337 images with 7.118 images collected at nighttime [46]. The dataset was recorded in 31 cities of 12 European countries.

The **NightOwls** dataset is specifically designed for pedestrian detection at night-time and includes the largest number of images collected at nighttime [26]. With over 279.000 images, its size is comparable to the **Caltech** dataset (daytime only). Due to larger variability in illumination and contrast, reduced color information, less visibility of pedestrians at nighttime, and additional reflections and high dynamics in the images, pedestrian detection at nighttime is more challenging than at daytime. The data was collected in several European cities, and the annotations include the additional attributes occlusion, difficulty, pose (back, front, left, right) as well as separate classes for pedestrian, cyclists and motorcyclists, and tracking identifiers. The

KAIST dataset is designed for research on sensor fusion based on a color camera and a thermal camera for pedestrian detection [48].

All the datasets differ in image resolution. The image resolution is 640 × 480 for Caltech and KAIST, 1392 × 512 for Kitti, 1024 × 640 for NightOwls, 1920 × 1024 for EuroCity, and 2048 × 1024 for CityPersons. Futhermore, the number of pedestrian annotations is interesting besides the total number of images: 9 k for Kitti, 31 k for CityPersons, 44 k (only nighttime) for NightOwls, 86 k (incl. 29 k at nighttime) for KAIST, 215 k (incl. 35 k nighttime) for EuroCity, and 289 k for Caltech. Most datasets include only daytime images. The NightOwls, EuroCity and KAIST dataset include nighttime images. Figure 10 shows challenging examples from the NightOwls dataset.

3.4 Evaluation Metrics for Pedestrian Detection

Pedestrian detectors are evaluated on a sufficiently large test dataset. Computing the evaluation metrics requires counting the occurrence of false detections (FP = false positives) and missing detections (FN = false negatives). Furthermore, the total number of ground truth annotations equals the sum of true detections (TP = true positives) and FN. The sum TP + FP equals all detections. Common measures for evaluating the performance of pedestrian detectors include:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Missrate = \frac{FN}{TP + FN} \tag{3}$$

The overlap between the bounding boxes of the ground truth and the detection is considered a TP, when their intersection-over-union (IoU) is larger than 50%. The larger the threshold for the IoU, e.g., 75%, the better the overlap between ground truth and detection:

$$IoU(GT, Detection) = \frac{Area_{GT} \cap Area_{Detection}}{Area_{GT} \cup Area_{Detection}} \tag{4}$$

Most commonly, the **Average Precision** (AP), **the Precision-Recall curve**, and the **Missrate-FPPI** (false positives per image) curve are reported in the literature as illustrated in Fig. 9. In the precision-recall curve, the Recall and the Precision is plotted for different thresholds s, where the detection score is larger than s for counting a detection as a TP. The AP is the area under the precision-recall curve. The higher the area under the curve (black line in Fig. 9), the better the detector.

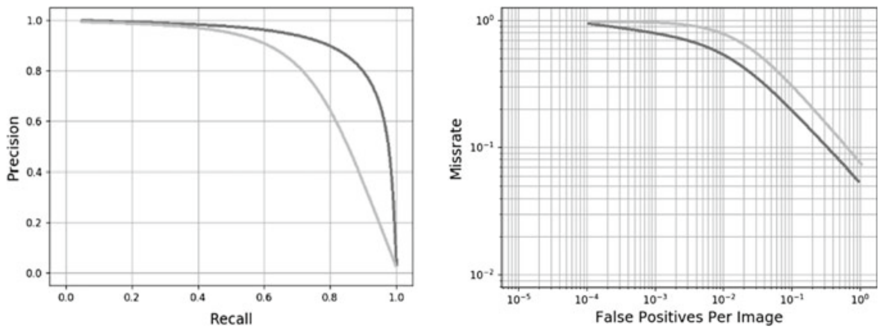


Fig. 9 Example for the Precision-Recall curve and the Missrate-FPPI curve

The average AP for several classes is called the mean AP (mAP) for multi-class classification. Results on the Missrate-FPPI curve are more intuitive to interpret in comparison to the Precision-Recall curve. Here, the lower the curve (black line), the better the detector.

Recently, the LRP (Localization Recall Precision) error has been proposed to take the localization accuracy better into account when computing the AP and to better distinguish between different detectors [49]. The LRP error compares the performance of detectors on a more granular level as the AP. Reason for this is that the common AP computation considers localization accuracy only indirectly (by the IoU computation for the TP, FP, and FN). Yet, localization accuracy is an important factor for many applications of object detection, including advanced driver assistance systems and automated driving.

The LRP error is a normalized, weighted sum over an IoU error for the TPs LRP_{IoU}, a measure for the false positives LRP_{FP} and a measure for the false negatives LRP_{FN}:

$$LRP(X, Y_s) = \frac{1}{Z}(w_{IoU} LRP_{IoU}(X, Y_s) + w_{FP} LRP_{FP}(X, Y_s) + w_{FN} LRP_{FN}(X, Y_s) \tag{5}$$

with the normalization $Z = |TP| + |FP| + |FN|$ for the ground truth boxes X and the detections Y_S with detection scores larger than a threshold s [49]. The weights are $w_{IoU} = |TP|/1 - \tau$, $w_{FP} = |Y_s|$, and $w_{FP} = |X|$. τ is the threshold for the IoU. The first component can be interpreted as a measure how close the bounding boxes of the ground truth and the valid detections are [49]:

$$LRP_{IoU}(X, Y_S) = \frac{1}{|TP|} \sum_{i=1}^{N_{TP}} (1 - IoU(X_i, Y_{xi})) \tag{6}$$

The second and third component are defined as [49]:

$$LRP_{FP}(X, Y_s) = 1 - Precision \tag{7}$$

$$LRP_{FN}(X, Y_s) = 1 - Recall \tag{8}$$

The optimal LRP (oLRP) can be used similar as the AP for comparing the performance of different detectors and is defined as [49]:

$$oLRP = \min_s LRP(X, Y_s) \tag{9}$$

In comparison to the AP metric, the oLRP provides an estimate for choosing the optimal threshold s for the detection score, includes an error estimate on the tightness of the bounding boxes between valid detections and ground truth, and can measure differences in detector performance, that may be hidden in the computation of the AP metric.

3.5 Discussion and Future Directions

Large improvement in pedestrian detection has been achieved over the last decade. This progress can be followed by comparing the Missrate for pedestrian detection over time.

Starting from a Missrate of 95% for the Viola and Jones algorithm (2004), and 68% for HOG-based classification (2005), the Missrate went down to 10% (2016) for CNN-based approaches on the Caltech dataset (Missrates are reported at 10^{-1} FPPI) [21, 24]. An estimation of the human baseline, i.e. the performance of humans correctly detecting pedestrians, for the Caltech dataset reports a Missrate of 6% [22]. Missrates of Faster-RCNN, R-FCN, SSD, and YOLO on the EuroCity dataset range between 8 and 10% for the deep network architectures [46]. Similar results are achieved for the CityPersons dataset with a Missrate of 13% for Faster-RCNN [45]. As detection at night is more challenging than detection at daytime, the Missrate of the NightOwls dataset for the Faster-RCNN is 19% [26].

Improvement in the extraction of informative features, learning meaningful representations, and granular classification helped achieving high detection rates for many traffic scene scenarios. When considering single-frame and camera-only pedestrian detection, the following challenges are future directions to further improve detection quality in difficult scenarios:

1. **Detection at Night**: Detecting pedestrians at night is difficult for various reasons. It is often hard to find appropriate camera settings for finding a balance between long exposure times and image gain for increasing the sensitivity. This results in sequences overlaid with motion blur or strong image noise. Furthermore, the dynamic of images recorded at night may exceed the dynamic range of a camera system. This leads to inhomogeneous illumination of the entire scene, including

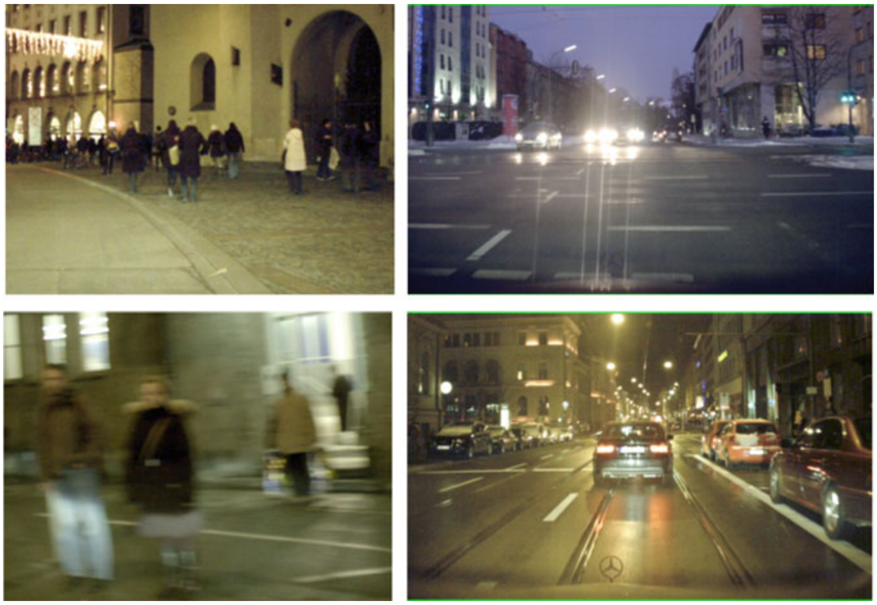


Fig. 10 Example images of the NightOwls dataset [26]

very dark and very bright areas illuminated by bright light sources or reflections. Additionally, occlusion can not only be caused by objects in the scene, but also by a lack of contrast for distinguishing people from the scene. The NightOwls dataset has been created to provide a dataset similar in size to the Caltech dataset for research on pedestrian detection at night [26]. Figure 10 shows challenging examples from the NightOwls dataset.

2. **Occlusion**: The evaluations on the Caltech, CityPersons, EuroCity, and NightOwls dataset are based on a reasonable subset of the annotations. Here, the term reasonable subset refers to a minimum height of the pedestrians (50 pixels) and only little occlusion. The Missrate increases significantly when including smaller or occluded pedestrians. The Missrate increases from 10 to 58% for the Faster R-CNN for the NightOwls dataset [26], and from 8 to 34% for the EuroCity dataset [46]. Occlusion influences detection performance strongly. Examples for occlusions are visualized in Fig. 11.
3. **Distant Objects**: Like occlusion, detection performance decreases significantly for distant and, hence, small pedestrians. Figure 12 shows how the informative content of the pixel representation decreases with resolution. For the NightOwls dataset, the Missrate is 35% for distant pedestrians, 8% for medium-scale pedestrians and 2% for large pedestrians [26]. For the EuroCity dataset, the Missrate increases from 8 to 17% when including small pedestrians in the evaluation [46]. Challenges for distant object detection are also reported for other objects such as vehicles [25, 27, 50].

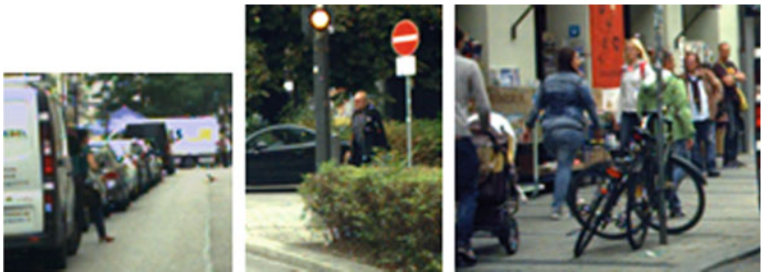


Fig. 11 Examples of occlusion where large fractions of a pedestrian's body is covered by other objects or pedestrians

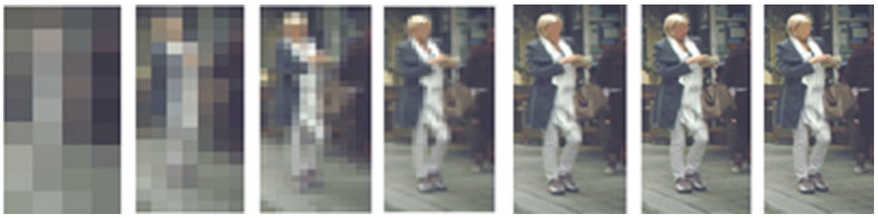


Fig. 12 Scale relates to the resolution of an object in the image. For small objects, the scale is low resulting in less informative features

4. **Localization**: Highly precise localization is important for pedestrian detection for advanced driver assistance systems and automated driving. Accurate localization can become difficult for rare poses, for crowds, or for occluded pedestrians.

Besides these challenges, future research directions in the area of pedestrian detection are panoptic segmentation, pose estimation, and mesh reconstruction, extending the scope of pedestrian detection which is based on bounding box representation today [51–56]. In addition to information such as position, size, and speed of pedestrians, new attributes like actions, gestures, intention, and emotions are required to further increase reliable detection and the detection performance for assisted and automated driving. Developing approaches for detecting the shape, skeleton, and motion besides bounding boxes representations for pedestrians can provide a meaningful contribution to more reliable pedestrian detection, see Fig. 13.

4 Summary and Outlook

Reliable pedestrian detection in single camera images is a crucial task for all advanced driver assistance systems and automated driving. The requirements towards pedestrian detection increase with increasing SAE level and come with very low miss rates and failure rates. While this article focuses on detecting pedestrians in single camera

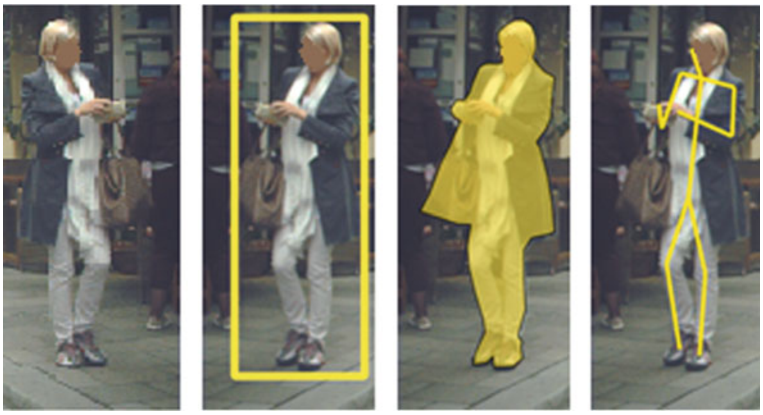


Fig. 13 Bounding box representation, shape representation, and skeleton representation

images using convolutional neural networks only, a variety of additional approaches can further increase the robustness of pedestrian detection in general.

Tracking pedestrians across several images can, when combined with road detection in general, significantly decrease the number of false positive and false negative detections and, hence, the miss and failure rates. Deep learning can be applied to tracking and several approaches have been proposed to solve tracking by CNNs [57–62]. The CNN learns appearance features of an object and localizes similar patches in the next frames, e.g., at 100 fps in [57], by estimating probability distributions in [58], by using a fully-convolutional Siamese network in [59], by learning domain-specific network branches which relate to training sequences in [60], and by combining weak trackers based on convolutional features with an online decision-theoretical Hedge algorithm to a strong tracker in [61].

Further information about the free space in a traffic situation or semantic segmentation can help to better distinguish between static and dynamic objects. Information about dynamic objects supports the identification of relevant pedestrians in crowds, potentially approaching the road and imposing a risk for collisions and accidents. Semantic segmentation is based on combining a convolutional network with a deconvolutional network [62–65]. Early works on the application of CNNs to semantic segmentation include [62, 63]. Semantic segmentation distinguishes between object classes, but not between instances. Instance segmentation segments the border of individual instances of cars or pedestrians [65]. Panoptic segmentation includes both segmentation of all objects in the image and instances for selected object classes [51, 52]. A review on semantic segmentation based on CNNs can be found in [65].

Additional sensors are required for reliably detecting pedestrians and vulnerable road users with increasing level of automation. Here, stereo cameras can provide depth information to better extract pedestrians from scattered background, or surround view camera systems that provide a holistic, 360° field of view around the vehicle. Radar and Lidar sensors can provide redundant data and can verify the detection of camera sensors to make pedestrian detection plausible. Deep learning approaches that can directly work on 3D data, collected with Lidar or Radar sensors, include VoxNet [66] and PointNet [67]. VoxNet is based on 3D CNNs and the network architecture consists of an occupancy grid as input, a set of convolutional layers for feature extraction, and fully connected layers for the final classification [66]. PointNet does not require voxelization and directly operates on raw point cloud data [67]. The classification network of PointNet contains two multi-layer perceptron layers within the feature computation and another one at the end of the network architecture for final classification [67].

When considering automated driving, initial approaches have been proposed that learn the driving behavior from sensor data only [68–75]. An example is PilotNet, which predicts the steering angle from camera data [68, 69]. The CNN architecture consists of five convolutional layers and three fully connected layers and imitates human driving. The decision of the network is based on observing both obvious features such as lane markings or other vehicles and subtle features such as bushes lining the edge of the road [69]. Long short-term memories (LSTM) can be included in the network design to model temporal dependencies for predicting the steering angle [70, 74, 75]. Conditional imitation learning is applied in [71] to overcome the limitations of [68–70], that the decision on where to drive is only based on sensor input. Conditional imitation learning can combine both navigational commands and imitating driving behavior based on camera input [71]. This is of advantage for taking decisions, e.g., at road junctions. This can be extended to generative adversarial imitation learning [73].

When considering partly or fully automated vehicle functions that can plan and execute specific driving operations such as stopping on crosswalks to let road users pass by, vehicle to pedestrian communication may be essential as well to confirm the road user that the vehicle has understood the situation and the street is safe to cross. For such applications, deep learning can play an additional role to support the interaction between humans and automated systems in the traffic. Applications in this field include observation and understanding of human behavior of humans inside the car, outside the car, and in the surrounding vehicles [76]. Examples are driver drowsiness detection, detection of driver's hand gestures, and activity recognition of pedestrians and bicycle drivers in the traffic [76, 77].

All these items, starting from single frame detection, over multi-sensor based detection approaches, and towards vehicle to pedestrian communication can build a highly reliable system for robust pedestrian detection at all levels of advanced driver assistance systems and automated driving. Deep learning plays a key role in the development of intelligent systems for observing the environment, fusing sensor data, and supporting decision making [2, 10, 16, 78–80]. The use of deep learning for safe and highly accurate pedestrian detection in traffic scenes is one of many applications of deep learning in the field of advanced driver assistance systems and automated driving.

References

1. Hinton, G.E., Osindero, S., Teh, Y.W.: Deep machine learning-a new frontier in artificial intelligence research. A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
2. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
3. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
4. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K.: Speed/accuracy trade-offs for modern convolutional object detectors. In: IEEE CVPR, vol. 4, July 2017
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
8. Hinton, G.E.: A practical guide to training restricted Boltzmann machines. In: Neural Networks: Tricks of the Trade, pp. 599–619. Springer, Berlin (2012)
9. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11**, 625–660 (2010)
10. Mohamed, A.R., Dahl, G.E., Hinton, G.: Acoustic modeling using deep belief networks. IEEE Trans. Audio Speech Lang. Process. **20**(1), 14–22 (2012)
11. Mhaskar, H.N.: Neural networks for optimal approximation of smooth and analytic functions. Neural Comput. **8**(1), 164–177 (1996)
12. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 1, pp. 886–893. IEEE, June 2005
13. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
14. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
15. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: AAAI, vol. 4, p. 12, February 2017
16. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A.P.: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471 (2016)
17. Russakovsky*, O., Deng*, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (2015) (* = equal contribution)
18. Mhaskar, H., Liao, Q., Poggio, T.A.: When and why are deep networks better than shallow ones? In: Association for the Advancement of Artificial Intelligence, pp. 2343–2349 (2017)
19. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., Sainath, T.: Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Process. Mag. **29** (2012).
20. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: a benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 304–311. IEEE, June 2009

21. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: an evaluation of the state of the art. IEEE Trans. Pattern Anal. Mach. Intell. **34**(4), 743–761 (2012)
22. Zhang, S., Benenson, R., Omran, M., Hosang, J., Schiele, B.: Towards reaching human performance in pedestrian detection. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 973–986 (2018)
23. Zhang, S., Benenson, R., Omran, M., Hosang, J., Schiele, B.: How far are we from solving pedestrian detection? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1259–1267 (2016)
24. Zhang, L., Lin, L., Liang, X., He, K.: Is faster R-CNN doing well for pedestrian detection? In: European Conference on Computer Vision, pp. 443–457. Springer, Cham, Oct 2016
25. Fattal, A.K., Karg, M., Scharfenberger, C., Adamy, J.: Distant vehicle detection: how well can region proposal networks cope with tiny objects at low resolution? In: 6th Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving, European Conference on Computer Vision (ECCV) (2018)
26. Neumann, L., Karg, M., Zhang, S., Scharfenberger, C., Piegert, E., Mistr, S., Prokofyeva, O., Thiel, R., Vedaldi, A., Zisserman, A., Schiele, B.: NightOwls: a pedestrians at night dataset. In: 14th Asian Conference on Computer Vision (ACCV) (2018)
27. Batzer, A.K., Scharfenberger, C., Karg, M., Lueke, S., Adamy, J.: Generic hypothesis generation for small and distant objects. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 2171–2178. IEEE, Nov 2016
28. Road Traffic, Destatis, Federal Statistical Office of Germany, Series 8, vol. 7 (2017)
29. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010)
30. Mao, J., Xiao, T., Jiang, Y., Cao, Z.: What can help pedestrian detection? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3127–3136 (2017)
31. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
32. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
33. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. arXiv:1612.08242 (2017)
34. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37. Springer, Cham, Oct 2016
35. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems, pp. 379–387 (2016)
36. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European Conference on Computer Vision, pp. 646–661. Springer, Cham, Oct 2016
37. Daya, I., Shafiee, M., Karg, M., Scharfenberger, C., Wong, A.: On Robustness of deep neural networks: a comprehensive study on the effect of architecture and weight initialization to susceptibility and transferability of adversarial attacks. In: 4th Annual Conference on Vision and Intelligent Systems (CVIS), Best Paper Award (2018)
38. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
39. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
40. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images, vol. 1, no. 4, p. 7. Technical report, University of Toronto (2009)
41. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: A mobile vision system for robust multi-person tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, pp. 1–8. IEEE, June 2008
42. Wojek, C., Walk, S., Schiele, B.: Multi-cue onboard pedestrian detection (2009)

43. Enzweiler, M., Gavrila, D.M.: Monocular pedestrian detection: survey and experiments. IEEE Trans. Pattern Anal. Mach. Intell. (12), 2179–2195 (2008)
44. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3354–3361. IEEE, June 2012
45. Zhang, S., Benenson, R., Schiele, B.: CityPersons: a diverse dataset for pedestrian detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, no. 2, p. 3, July 2017
46. Braun, M., Krebs, S., Flohr, F., Gavrila, D.M.: The EuroCity persons dataset: a novel benchmark for object detection. arXiv:1805.07193 (2018)
47. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: a multimodal dataset for autonomous driving. arXiv:1903.11027 (2019)
48. Hwang, S., Park, J., Kim, N., Choi, Y., So Kweon, I.: Multispectral pedestrian detection: benchmark dataset and baseline. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1037–1045 (2015)
49. Oksuz, K., Cam, B.C., Akbas, E., Kalkan, S.: Localization recall precision (LRP): a new performance metric for object detection. In: European Conference on Computer Vision (ECCV), vol. 6, July 2018
50. Fattal, A.K., Karg, M., Scharfenberger, C., Adamy, J.: Saliency-guided region proposal network for CNN based object detection. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 1–8. IEEE, Oct 2017
51. Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P.: Panoptic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
52. Kirillov, A., Girshick, R., He, K., Dollár, P.: Panoptic feature pyramid networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
53. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision, pp. 483–499. Springer, Cham, Oct 2016
54. Tung, H.Y., Tung, H.W., Yumer, E., Fragkiadaki, K.: Self-supervised learning of motion capture. In: Advances in Neural Information Processing Systems, pp. 5236–5246 (2017)
55. Pavlakos, G., Zhu, L., Zhou, X., Daniilidis, K.: Learning to estimate 3D human pose and shape from a single color image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 459–468 (2018)
56. Insafutdinov, E., Dosovitskiy, A.: Unsupervised learning of shape and pose with differentiable point clouds. In: Advances in Neural Information Processing Systems, pp. 2802–2812 (2018)
57. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 FPS with deep regression networks. In European Conference on Computer Vision, pp. 749–765. Springer, Cham, Oct 2016
58. Zhai, M., Chen, L., Mori, G., Javan Roshtkhari, M.: Deep learning of appearance models for online object tracking. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 0–0 (2018)
59. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: European Conference on Computer Vision, pp. 850–865. Springer, Cham, Oct 2016
60. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4293–4302 (2016)
61. Qi, Y., Zhang, S., Qin, L., Yao, H., Huang, Q., Lim, J., Yang, M.H.: Hedged deep tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4303–4311 (2016)
62. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1520–1528 (2015)
63. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)

64. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2359–2367 (2017)
65. Guo, Y., Liu, Y., Georgiou, T., Lew, M.S.: A review of semantic segmentation using deep neural networks. Int. J. Multimed. Inf. Retr. **7**(2), 87–93 (2018)
66. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660 (2017)
67. Maturana, D., Scherer, S.: Voxnet: a 3D convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 922–928. IEEE, Sept 2015
68. Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., Muller, U.: Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv:1704.07911 (2017)
69. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X.: End to end learning for self-driving cars. arXiv:1604.07316 (2016)
70. Fernando, T., Denman, S., Sridharan, S., Fookes, C.: Going deeper: autonomous steering with neural memory networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 214–221 (2017)
71. Codevilla, F., Miiller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–9. IEEE, May 2018
72. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2722–2730 (2015)
73. Kuefler, A., Morton, J., Wheeler, T., Kochenderfer, M.: Imitating driver behavior with generative adversarial networks. In: 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 204–211. IEEE, June 2017
74. Xu, H., Gao, Y., Yu, F., Darrell, T.: End-to-end learning of driving models from large-scale video datasets. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2174–2182 (2017)
75. Morton, J., Wheeler, T.A., Kochenderfer, M.J.: Analysis of recurrent neural networks for probabilistic modeling of driver behavior. IEEE Trans. Intell. Transp. Syst. **18**(5), 1289–1298 (2017)
76. Ohn-Bar, E., Trivedi, M.M.: Looking at humans in the age of self-driving and highly automated vehicles. IEEE Trans. Intell. Veh. **1**(1), 90–104 (2016)
77. Reddy, B., Kim, Y.H., Yun, S., Seo, C., Jang, J.: Real-time driver drowsiness detection for embedded system using model compression of deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 121–128 (2017)
78. Lemley, J., Bazrafkan, S., Corcoran, P.: Deep learning for consumer devices and services: pushing the limits for machine learning, artificial intelligence, and computer vision. IEEE Consum. Electron. Mag. **6**(2), 48–56 (2017)
79. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
80. Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., Lew, M.S.: Deep learning for visual understanding: a review. Neurocomputing **187**, 27–48 (2016)

Deep Learning in Speaker Recognition

Omid Ghahabi, Pooyan Safari and Javier Hernando

Abstract It is supposed in Speaker Recognition (SR) that everyone has a unique voice which could be used as an identity rather than or in addition to other identities such as fingerprint, face, or iris. Even though steps have been taken long ago to apply neural networks in SR, recent advances in computing hardware, new deep learning (DL) architectures and training methods, and access to a large amount of training data have inspired the research community to make use of DL as in a large variety of other signal processing applications. In this chapter, the traditional principle techniques in SR are first briefly reviewed and the potential signal processing aspects of these techniques which can be improved by DL are addressed. Then the recent most successful DL architectures used in SR are introduced and some illustrative experiments from the authors are included.

Keywords Speaker recognition · Deep learning · Speaker verification · Speaker embedding · Deep neural network

1 Introduction

Speaker Recognition (SR) is the task of automatically recognizing the speaker of an utterance either in an identification or verification fashion. All the recognition process is only based on the speech signals captured from the speakers. In other words, it is supposed that everyone has a unique voice which could be used as an identity. Although the research steps to use neural networks in SR have been taken

O. Ghahabi (✉)
EML European Media Laboratory GmbH, Heidelberg, Germany
e-mail: omid.ghahabi@eml.org

P. Safari · J. Hernando
Universitat Politècnica de Catalunya - BarcelonaTech, Barcelona, Spain
e-mail: pooyan.safari@tsc.upc.edu

J. Hernando
e-mail: javier.hernando@upc.edu

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_6

long ago (e.g., [1–13]), recent advances in computing hardware, new Deep Learning (DL) architectures, and access to a large amount of training data have encouraged the research community to use DL again.

DL strategies can be used in the frontend of a SR system. A possible use is in the state-of-the-art i-vector [14]–a compact representation of characteristics of a speech signal which is widely used in not only SR but also in other recognition tasks like speech, language, and emotion. Deep Neural Networks (DNNs) have been used in the i-vector extraction algorithm for two main purposes. Firstly, a DNN is used as an acoustic model rather than the typical Gaussian Mixture Model (GMM) [15–19]. Secondly, traditional spectral features are substituted or concatenated with the so-called bottleneck features [18, 20]. A notable accuracy gain can be obtained in both cases but it is reported that the use of a Gaussian acoustic model trained on the concatenation of bottleneck and spectral features leads usually to higher quality i-vectors [18, 20]. However, the main drawbacks are the increase of the computational cost for i-vector extraction and the requirement of phonetic labels for DL acoustic modeling.

Another possible use of DL in the frontend is to represent the speaker characteristics of a speech signal with a single low dimensional vector using a DL architecture, rather than the traditional i-vector algorithm. These vectors are often referred to as speaker embeddings. In this chapter, we divide them into two main groups of supervised and unsupervised speaker embeddings. The first group is the case for which the DL architecture is usually trained given the speaker-labeled background data. Typically, the inputs of the neural network are a sequence of feature vectors and the outputs are speaker classes. Different architectures, activation functions, and training procedures have been proposed (e.g., [21–24]). The experimental results have shown that, in most cases, the bigger improvements are obtained on the shorter signals compared to the traditional i-vectors [23, 24], which implies that DL technology can model the speaker characteristics of a short-duration speech signal better than the traditional signal processing techniques. This is important for real-world applications where the decision should be made in very few seconds, provided that the computational cost is still reasonable. It is recently shown that data augmentation can notably improve the performance of speaker embeddings and make them competitive with traditional i-vectors even for long-duration speech segments [25]. Nevertheless, the need of speaker labels for training the network is one of the disadvantages of these techniques. Moreover, speaker embeddings extracted from hidden layer outputs are not so compatible with Probabilistic Linear Discriminant Analysis (PLDA) [26], the most effective backend in SR, because the posterior distribution of hidden layer outputs are usually not truly Gaussian. The background data in the second group is free of any kind of labels, which can be considered one of the advantages of these techniques. The unsupervised techniques used for this purpose try usually to reconstruct the input and to minimize the reconstruction error or the cross entropy during the training process. Typically, a kind of adaptation to the speaker data of each utterance is performed in these techniques. For instance, in [27, 28] Maximum a Posteriori (MAP) adapted GMM supervectors are given to the network as input vectors and in [29], the parameters of a universal network are adapted to the input sequence of feature vectors of each speaker. As for supervised speaker embeddings, these vectors suffers also from incompatibility with PLDA backend. The authors have tried to solve this problem by introducing a variant of Rectified Linear Unit (ReLU) referred to as Variable ReLU (VReLU) [28].

One of the most effective backend techniques for i-vectors is PLDA [26, 30], which performs the scoring along with the session variability compensation. Usually, a large number of different speakers with several speech samples each are necessary for PLDA to work efficiently. Access to the speaker labeled data is costly and in some cases almost impossible. Moreover, the amount of performance gain, in terms of accuracy, for short utterances is not as much as that for long utterances. These facts motivated the research community to look for DL based alternative backends to PLDA. Several techniques have been proposed. Most of these approaches use the speaker labels of the background data for training, as in PLDA, and mostly with no significant gain compared to PLDA. For example, different combinations of Restricted Boltzmann Machines (RBMs) have been proposed in [31, 32] to classify i-vectors and in [96] to learn speaker and channel factor subspaces in a PLDA simulation. RBMs in [33] and DNNs in [34] are used to increase the discrimination power of i-vectors given speaker-labeled background data. A nonlinear PLDA is simulated in [35] using a tied variational autoencoder architecture. In [36, 37], a combination of RBM, autoencoder, and PLDA is proposed for speaker and channel variability compensation, which shows some improvements compared to using only PLDA.

Recently, a SR challenge was organized by the National Institute of Standard and Technology (NIST) [38] to address how a comparable performance with PLDA can be achieved when the development data is not labeled. Although the use of unsupervised automatic labeling algorithms were proposed by some participating teams [39, 40], those algorithms cannot correctly estimate all the labels. Additionally, it is supposed that several samples are available for each speaker in the background data which could not be true in reality. The authors proposed a hybrid Deep Belief Network (DBN)-DNN architecture in [41–43] to address this problem, filling the performance gap between cosine and PLDA scoring when no speaker-labeled development data is available. The proposed architecture is initialized with speaker-specific parameters and tries to discriminate between target and selected non-target, known also as impostor, speakers in the i-vector space. The experiments on the challenge database showed the excellent results of the proposed architecture alone and by the combination with unsupervised automatic labeling techniques.

A natural choice for DL is to train an end-to-end SR system, capable of doing multiple stages of data processing with a unified network. End-to-end architectures were successfully applied in many other tasks such as [44, 45]. In SR, there have been several attempts to build such models as e.g., proposed in [46], where speaker models take speaker spectral features as input and output the similarity scores. However, these architectures are not yet so competitive with the methods based on vector representation of speakers.

The rest of the chapter is organized as follows. Section 2 reviews briefly the conventional techniques used in a typical state-of-the-art speaker recognition system from feature extraction to classification and scoring. Section 3 describes how DL has been used in a frontend of a SR system. It will mainly talk about the use of DL technology in i-vector extraction, supervised and unsupervised speaker embeddings and will summarize the works of the authors proposed in [28, 29]. Section 4 summarizes the recent research works on the use of DL as a backend in a SR system and then will mainly describe briefly the work of the authors presented in [43]. Section 5 describes how both frontend and backend could be performed at once with a DL architecture, i.e., giving the output scores given the input feature vectors. Finally, Sect. 6 summarizes the conclusions and gives some hints for the future work.

2 Conventional Techniques

Recognizing the identity of individuals only by their voice, known as SR, dates back around five decades. Usually, the speech acoustic features are used to discriminate between different speakers. Acoustic features reflect both physical, e.g., the size and the shape of the throat and mouth, and behavioral characteristics, like the voice pitch and the speaking style, of an individual. Two main branches are usually considered for speaker recognition, namely identification and verification. Speaker identification can be seen as a multi-class classification task where a test utterance has to be assigned necessarily to one of the enrolled speakers (closed-set) or could be identified also as an unknown speaker (open-set). An open-set speaker identification task is usually more difficult since a robust decision threshold should be defined as well. Speaker Verification (SV), on the other hand, can be seen as a two-class classification task where a claimed identity from an unknown speaker should be verified by the system whether the unknown speaker own the claimed identity. SR can be text-dependent or text-independent. The text-dependent SV requires the speaker saying exactly a given text, password, or sequence of numbers, whereas the text-independent SV is based on free speech. In principle, the text-dependent task is more accurate and needs less amount of training and testing data compared to the text-independent one. However, the text-independent speaker verification is more convenient for users as they can speak freely. Moreover, in some applications only the text-independent task is applicable. In this chapter, we mainly focus on the text-independent SV task.

As it is shown in Fig. 1, SV involves two main stages: the training phase in which the target speakers are enrolled and the testing phase in which the claimed identity of the unknown speaker is verified. It is possible to model the speakers either in a generative way such as GMM [47] which estimates the distribution of feature vectors within each speaker, or in a discriminative fashion such as Support Vector Machine (SVM) [48] and DNN which model the boundary between speakers. In the following, we summarize the main parts of a state-of-the-art SV system.

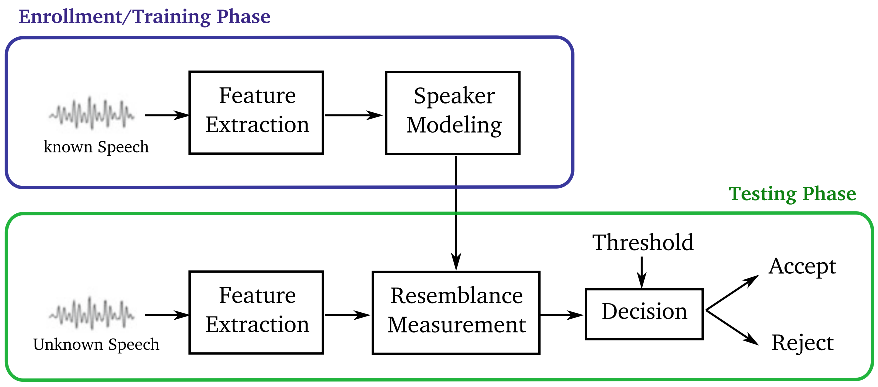


Fig. 1 Block diagram representation of a basic SV system

2.1 Feature Extraction

Feature extraction is usually one of the primary steps in any kind of pattern recognition system. It is the process of extracting meaningful features from the raw data. In speech processing, features vectors obtained from a speech signal contain usually the acoustic characteristics, speaker, and language information. Depending on the application and the task, the recognition system will focus only on the target information, which is speaker in this case. One of the standard features in SR is a set of short-term acoustic features obtained from the speech spectrum. The spectrum of a speech signal reflects usually the physiology of the vocal tract which is an important factor for discriminating between speakers. Currently, the most commonly in use feature is Mel-Frequency Cepstral Coefficient (MFCC) [49], which has shown a good performance also in other speech processing tasks. Apart from MFCCs, other features like Linear Predictive Coefficient (LPC), Linear Frequency Cepstral Coefficient (LFCC), Perceptual Linear Predictive (PLP) coefficients, and frequency filtered filter-bank energies or in short Frequency Filtering (FF) coefficients [50, 51] are used as well. In SR, the first and the second order time derivatives known as delta and delta-delta coefficients are usually obtained to assist the recognition. The delta energy is also commonly added to feature vectors.

Feature normalization strategies are also employed for environmental mismatch compensation. Typically, the mean of the cepstral coefficients is removed in order to avoid nonlinear effects due to the session variability. Optionally, the variance of the cepstral coefficients can be also normalized to unit over a sliding window or over the whole utterance. Sometimes, the shape of the cepstral coefficient distribution is also taken into consideration. Among the several techniques, which have been proposed for this purpose, Cepstral Mean Normalization (CMN), Cepstral Mean and Variance Normalization (CMVN), and feature warping are more commonly in use [52].

2.2 Supervectors and i-Vectors

The traditional speaker modeling is based on GMMs. A GMM is a weighted sum of M Gaussian densities defined by three sets of parameters as $\lambda = \{w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{M}$, where w_i, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are, respectively, the weight, the mean vector, and the covariance matrix of the ith Gaussian density. The GMM parameters are estimated using the Expectation-Maximization (EM) algorithm as in [47]. As the amount of the enrollment data for each speaker is usually few, it is not so efficient to train a GMM for each speaker from scratch. Therefore, a global GMM, which is referred to as Universal Background Model (UBM), is first trained using a large number of utterances, and then the UBM is adapted to a few amount of data of each speaker [53]. The adaptation is typically performed using the MAP estimation given in [53]. GMM supervectors are obtained by concatenating the D-dimensional mean vectors of the M-mixture adapted GMM [54]. For the speaker a, a GMM supervector is represented as,

$$\boldsymbol{s}^a = (\boldsymbol{\mu}_1^a, \boldsymbol{\mu}_2^a, ..., \boldsymbol{\mu}_M^a)^t \tag{1}$$

where t refers to a transpose operation.

It is possible to compare two supervectors based on their distance, but it is commonly more efficient to discriminate them by SVMs, leading to a hybrid generative-discriminative classifier [54–56].

As the train and test speech utterances are usually spoken in different sessions, e.g., the use of different microphones or different channels for transferring the speech signal, some session variability compensation techniques are required, in addition to the compensation techniques in the feature domain as described in Sect. 2.1, for having a higher recognition accuracy and a more robust system. Two commonly in use session compensation techniques in the supervector domain are the Nuisance Attribute Projection (NAP) [57, 58] and Within-Class Covariance Normalization (WCCN) [59].

Another successful technique for session variability compensation of supervectors is the Joint Factor Analysis (JFA) [60] which models the supervectors as a linear combination of the speaker and channel components. However, as in SVM based techniques, the session variability compensation is carried out in the supervector domain which is very high dimensional. Therefore, a big memory space is required for the training of the compensation matrices and it is computationally expensive. It is proposed in [14] to first reduce the dimension of the supervectors through an effective factor analysis technique and then to perform session compensation in the lower dimensional space. It is supposed that the supervector $\boldsymbol{s}^a$ can be modeled as follows [14],

$$\boldsymbol{s}^a = \boldsymbol{s}^{ubm} + \boldsymbol{T}\boldsymbol{\nu} \tag{2}$$

where $\boldsymbol{s}^{ubm}$ is a mean supervector obtained from UBM, $\boldsymbol{T}$ is the total variability matrix, and $\boldsymbol{\nu}$ is a vector of latent variables. The mean vector of the posterior distribution of $\boldsymbol{\nu}$, conditioned on the Baum-Welch statistics of the given speech utterance, is referred to as i-vector $\boldsymbol{\omega}$ and computed as follows,

$$\boldsymbol{\omega} = \left(\boldsymbol{I} + \boldsymbol{T}^t \boldsymbol{\Sigma}^{-1} \mathcal{N}(\boldsymbol{u}) \boldsymbol{T}\right)^{-1} \boldsymbol{T}^t \boldsymbol{\Sigma}^{-1} \tilde{\mathcal{F}}(\boldsymbol{u}) \tag{3}$$

where $\mathcal{N}(\boldsymbol{u})$ is a diagonal matrix containing the zeroth order Baum-Welch statistics, $\tilde{\mathcal{F}}(\boldsymbol{u})$ is a supervector of the centralized first order statistics, and $\boldsymbol{\Sigma}$ is a diagonal covariance matrix initialized by $\boldsymbol{\Sigma}_{ubm}$ and updated during the factor analysis training. The $\boldsymbol{T}$ matrix is trained using the EM algorithm given the Baum-Welch statistics from the development data. More details can be found in [14].

2.3 *i-Vector Scoring*

The preliminary scoring technique for i-vectors is cosine distance [14, 61],

$$score^{(cosine)}\left(\boldsymbol{\omega}_1, \boldsymbol{\omega}_2\right) = \frac{\boldsymbol{\omega}_1^t \boldsymbol{\omega}_2}{\|\boldsymbol{\omega}_1\| \times \|\boldsymbol{\omega}_2\|} \tag{4}$$

where $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are the target and the test i-vectors and $\|\boldsymbol{\omega}\|$ denotes the norm of the i-vector $\boldsymbol{\omega}$ computed as $\sqrt{\omega_1^2, \omega_2^2, ..., \omega_n^2}$.

If the speaker labels for the background speech utterances are not available, the cosine scoring gives as such a reasonable accuracy. However, given the speaker labels it is more effective if a session variability compensation technique is applied before scoring. Linear Discriminant Analysis (LDA), WCCN, or a combination of them is usually used [14]. It should be noted that speaker labels are costly and are not always accessible. PLDA [26] is a more effective technique when speaker labels are available for the background data. In PLDA, scoring is performed along with the session variability compensation. It assumes that each i-vector can be decomposed as,

$$\boldsymbol{\omega} = \boldsymbol{m} + \boldsymbol{\Phi}\zeta + \varepsilon \tag{5}$$

where $\boldsymbol{m}$ is a global mean vector, $\boldsymbol{\Phi}$ represents the eigenvoices, ζ is a latent vector with a normal distribution prior, and ε is the residual vector normally distributed with zero mean and the full covariance matrix $\boldsymbol{\Sigma}$. The parameters of the model are estimated by the EM algorithm given a large amount of speaker-labeled development data [26]. Between and within class i-vector covariance matrices are stored and used for scoring. More details can be found in [62].

It is shown [62] that the length normalization ($\boldsymbol{\omega} \leftarrow \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}$) helps the Gaussianity of i-vectors which leads to a comparable performance to a more complicated PLDA, which is referred to as heavy-tailed PLDA [30, 63]. As proposed in [64], i-vectors, in an i-vector baseline system, are first globally whitened as,

$$\boldsymbol{\omega}' = \boldsymbol{H}\boldsymbol{\omega} \tag{6}$$

$$\boldsymbol{H} = \boldsymbol{V}\left(\boldsymbol{D} + \epsilon\right)^{-1/2} \boldsymbol{V}^t \tag{7}$$

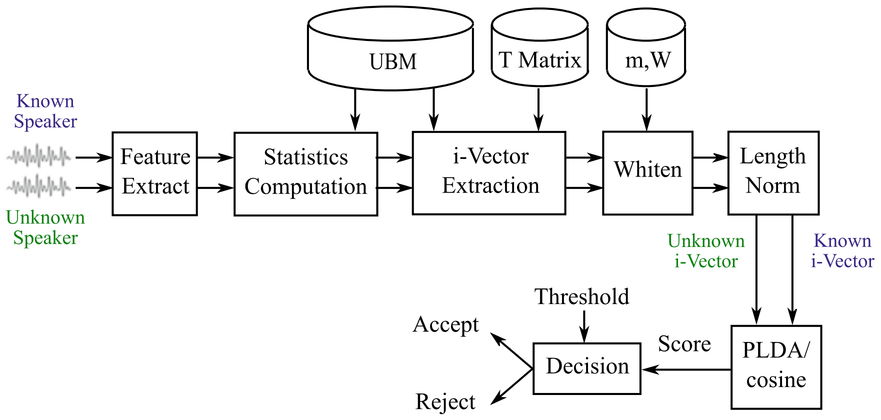


Fig. 2 Block diagram of a typical i-vector/PLDA SV system

where $\boldsymbol{H}$ is the whitening matrix, $\boldsymbol{V}$ is the matrix of eigenvectors obtained on the covariance matrix of the background i-vectors, $\boldsymbol{D}$ is the diagonal matrix of the corresponding eigenvalues, and ϵ is a very small constant regularization factor. After whitening, i-vectors are length-normalized. The block-diagram of Fig. 2 summarizes a typical conventional state-of-the-art system for SV. More details regarding the conventional techniques can be found in [65–67].

3 Deep Learning Frontends

We split the frontend into three subsections. The first subsection is about how DL can be used in an i-vector extraction process from bottleneck features to DL based acoustic modeling. There will be two subsections for supervised and unsupervised speaker embeddings. The embeddings are referred to the stand-alone speaker representations which are built independent from i-vector process and can be used directly in recognition tasks.

3.1 Feature and i-Vector Extraction

The traditional i-vector approach consists in three main stages: Baum-Welch statistics collection, i-vector extraction, and PLDA backend. Recently, it is shown that if the Baum-Welch statistics are computed with respect to a DNN rather than a GMM or if bottleneck features are used in addition to conventional spectral features, a substantial improvement can be achieved [15, 16, 18].

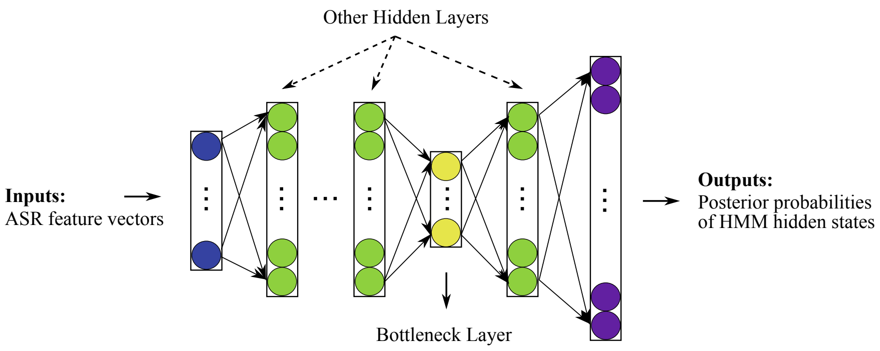


Fig. 3 A typical DNN architecture used for acoustic modeling and bottleneck feature extraction in DNN based i-vector approach

A variant of DL architectures have been used for acoustic modeling. Figure 3 shows a typical DNN architecture used for both Baum-Welch statistics computation and bottleneck feature extraction. The network is preliminary trained for acoustic modeling in Automatic Speech Recognition (ASR).

The output layer represents the acoustic classes, which are typically the states of the Hidden Markov Model (HMM) in ASR (triphones). The input layer takes usually a concatenation of successive ASR feature vectors. ASR feature vectors are usually the log filter bank energies without any delta or delta delta coefficients. The activation function for the output layer is softmax, for the bottleneck layer is usually linear, and for other hidden layers can be sigmoid, rectified linear, or other similar functions like tanh.

Given the DNN acoustic model, the zeroth and the first order Baum-Welch statistics are computed as follows,

$$\mathcal{N}_k(\boldsymbol{u}) = \sum_t p(o_k|x_t) \tag{8}$$

$$\mathcal{F}_k(\boldsymbol{u}) = \sum_t p(o_k|x_t)\hat{x}_t \tag{9}$$

where $\mathcal{N}_k(\boldsymbol{u})$ and $\mathcal{F}_k(\boldsymbol{u})$ are, respectively, the zero and first order statistics given the utterance $\boldsymbol{u}$, $p(o_k|x_t)$ is the posterior probability of kth output unit given the ASR feature vector x_t, and $\hat{x}_t$ is the speaker feature vector which can differ from x_t. Given the Baum-Welch statistics, the $\boldsymbol{T}$ matrix training and the i-vector extraction process will be the same as with GMM acoustic model. However, despite the GMM model, the non-speech frames have been also used in the training of the DNN acoustic model. Therefore, there will be two possible options for DNN statistics computation as proposed in [37]. The first option is to use an external Voice Activity Detection (VAD), discard non-speech frames, and use only the DNN output units corresponding to the speech states in HMM. The second option is not to use any external VAD, compute the statistics for all frames, and, like in the first option, to discard the output probabilities which correspond to non-speech states. The second option can be interpreted as a kind of a soft VAD rather than the hard VAD in the first option. It has been shown that the second option leads usually to a better performance in terms of the accuracy [68, 69]. It is worth noting that for both options, the zeroth order statistics should be normalized by the sum over the output units corresponding to speech states.

Although the i-vector extraction using DNN acoustic models leads to a higher accuracy in general, there are also some disadvantages. First of all, the use of DNN itself increases the computational cost of the statistics to a great extent. Moreover, the number of the output units is usually much higher than the number of Gaussian mixtures in the GMM. This means that the dimensions of supervectors will be much higher than the dimensions of GMM supervectors leading to higher computational cost in both $\boldsymbol{T}$ matrix training and i-vector extraction. Additionally, the language dependency of DNN acoustic models, the use of two different feature vectors for the computation of the zeroth and the first order statistics (e.q., Eqs. 8 and 9), and the need of the phonetic labels for training the DNN acoustic model are other shortcomings of DNN based i-vector extraction approaches.

On the other hand, DNNs have been used to extract the so-called bottleneck features. As it is shown in Fig. 3, the hidden layer before the last hidden layer is usually much smaller than the other hidden layers and considered as the bottleneck layer. The hidden unit values of this layer, given the input vectors, are referred to as bottleneck features. However, these features are usually highly correlated and, therefore, they need some decorrelating, typically using Principal Component Analysis (PCA), before usage. The traditional spectral features can be replaced or concatenated with DNN bottleneck features and then a GMM or a DNN background model can be used to compute the statistics [18, 20, 70]. Alternatively, other DL architectures like Convolutional Neural Network (CNN) have been also employed to build an acoustic model or to produce bottleneck features [71].

The recently proposed Adversarial Networks (ANs) [72] have drawn attentions in the machine learning community. They composed of two competing networks of which one (generator) tries to fool the other (discriminator). They have been applied to audio and speech applications such as in [73, 74]. For SR task they have been employed to produce bottleneck features in [75]. The input to the AN is the speaker spectral features and the outputs of the Encoding Network (EN) are considered as bottleneck features. For the training of the Discriminative Network (DN), the noise types or clean speech are used as the output labels while for the training of the EN the output label is always considered as clean speech careless to the input. While training the EN, the DN parameters are fixed and vice versa. With this architecture, the EN learns to produce features which are invariant to noise types.

3.2 Supervised Speaker Embeddings

Recently, there have been several attempts to apply DL in order to build speaker embeddings. Speaker embedding is often referred to a single low dimensional vector representation of a speaker extracted using a neural network. Supervised speaker embeddings are produced by training a deep architecture, using speaker-labeled background data. This network, which is capable to produce high-level features, is usually trained to discriminate the background speakers. Then in the testing phase, the output layer is discarded, the feature vectors of an unknown speaker are given through the network, and usually the average or the weighted average of the activation of a given hidden layer are considered as the speaker embedding [21, 23].

Although several supervised speaker embeddings have been proposed, two of them are more commonly in use, namely d-vector [21] and x-vector [24, 25]. Figure 4 shows a general architecture of a d-vector extraction training network. The inputs of the network are the speaker feature vectors stacked over a context window, e.g., the feature vector of a speech frame in time t is stacked with the feature vectors of the frames in times $t-2$, $t-1$ and $t+1$, $t+2$ covering a context window of 5 frames. In the training phase, the network tries to discriminate background speakers in the frame level. In the testing phase, the output layer is discarded and the average of the activation of the last hidden layer is usually considered as the d-vector for the given speaker. Although the proposed network in [21] is a maxpooling DNN and the hidden units in the last two hidden layers are dropped out by 50%, any deep architecture like CNN or Long Short-Term Memory (LSTM) can also be used (e.g., [23, 76]).

Figure 5 shows a typical architecture for x-vector extraction. The idea behind is similar to d-vector but the network for x-vector extraction tries to discriminate background speakers in the segment level rather than the frame level in d-vector

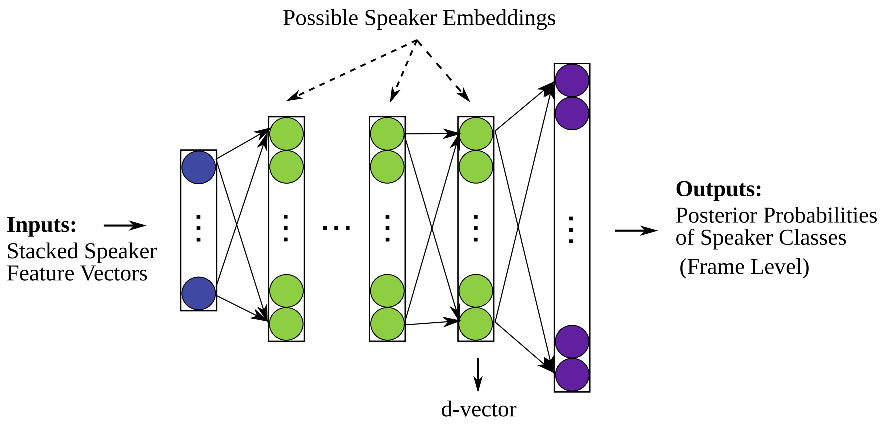

Possible Speaker Embeddings
Inputs:
Stacked Speaker Feature Vectors
Outputs:
Posterior Probabilities of Speaker Classes
(Frame Level)
d-vector

Fig. 4 A general architecture for d-vector extraction. The network tries to discriminate between background speakers in the frame level. The output layer is discarded for d-vector extraction in the testing phase

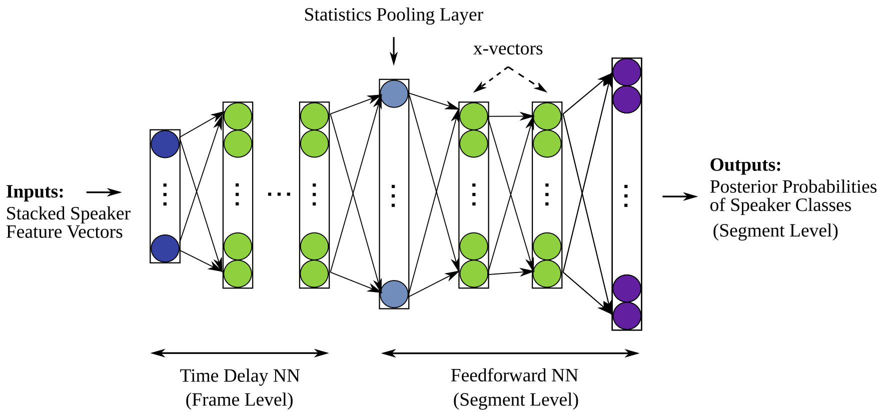


Fig. 5 A general architecture for x-vector extraction. The network tries to discriminate between background speakers in the segment level. The output layer is discarded for x-vector extraction in the testing phase

extraction. As it is shown in Fig. 5, the network is composed of two main parts. The first part is a Time Delay Neural Network (TDNN) which works in the frame level and takes the stacked feature vectors as the input. The second part is a feedforward neural network which works in the segment level and takes the concatenated vectors of the mean and the standard deviation of the activations in the last hidden layer of the first part, i.e., the TDNN. There is a statistics pooling layer in the middle as a connection between the first and the second parts of the network. The statistics pooling layer aggregates over the variable-length input segments and prepares the fixed-dimensional statistics vectors as the inputs to the second part of the network. The TDNN part of the network proposed in [24] is composed of 4 hidden layers. The input feature vectors are stacked in a 5-frame context window. The first two hidden layers consider a temporal context of the previous layer with configurations $\{t-2, t, t+2\}$ and $\{t-3, t, t+3\}$, respectively. In the next two layers, no context is added. In total, the first part of the network covers a temporal context of $t-8$ to $t+8$. The second part of the network has only two hidden layers which their activations can be used as speaker embeddings or x-vectors in this architecture. The primarily results showed that x-vectors outperform the traditional i-vectors only for short duration speech segments [24]. However, a recent work has shown that data augmentation, consisting of added noise and reverberation, can significantly improve the performance of x-vectors while it is not that effective for i-vectors [25]. There have also been some efforts to improve the quality and generalization powers of x-vectors like by the modification of the network architecture [77] and the training procedure [77–80].

Inspired by the triplet loss employed for image recognition in [81–83], it is also used as a training procedure in a speaker embedding extraction architecture rather than discrimination among background speakers as used in d-vector and x-vector extraction networks (Figs. 4 and 5) [84, 85]. The triplet loss function is defined on triplets of speech segments namely anchor, positive, and negative samples. Given an utterance of the anchor speaker, the goal is to reduce its distance from other utterances of the same speaker (positive) and increase its distance from the utterances of any other speakers (negative). In other words, the loss function tries to minimize the intra-speaker variability while maximize the inter-speaker variability at once. The most recent works have focused mainly on different deep architectures like ResNet [86] or VGG [87] as for instance in [88–90] and the practical aspects of speaker embeddings like generalization power and speed [91].

In all of the mentioned works above, the input speaker feature vectors to the neural network have been the most commonly in use hand-crafted features like MFCCs and Filter Bank Energys (FBEs). There have recently been some attempts to get rid of these features and work directly on raw speech samples [92, 93]. However, it is still very new and under development and investigation.

3.3 Unsupervised Speaker Embeddings

All the speaker embeddings so far need speaker labels of the background data in one way or another. In supervised techniques, usually the performance is better since more information are fed into the system, which are mostly the labels of the background data. However, extracting the speaker labels are usually expensive and not a trivial task. This motivates the methods based on unsupervised techniques. Having good representational power, makes RBMs useful candidates for this purpose. They are computationally low cost and unsupervised. In SR, they have been used for different purposes such as i-vector classification [94, 95], speaker factors [96], and feature extraction [97]. They have been employed in an adaptation process [41, 42, 46, 98] to further discriminatively learn target and impostor speaker models. RBMs were also used to pre-train DBNs in order to extract Baum-Welch statistics [16, 99]. There have been also a few attempts to create alternative vector-based speaker representation using RBMs [27–29, 100].

In the following, we will mainly describe the work of the authors proposed in [28, 29]. In [28], GMM supervectors are considered as the inputs to the RBM and activations of the hidden layer could be the low dimensional speaker embedding vectors. An RBM, which was referred to as Universal RBM (URBM), is trained using the background GMM supervectors. URBM tries to learn the total session and speaker variability among background supervectors. Once the URBM is trained, the matrix of the visible-hidden connection weight parameters is used for dimension reduction of unseen GMM supervectors. These low-dimensional vectors are referred to as GMM-RBM vectors in that work. Different hidden units and transformation functions have been tried for URBM training and extraction of GMM-RBM vectors. An efficient activation function which is referred to as variable RELU (VReLU) has also been proposed as follows [28],

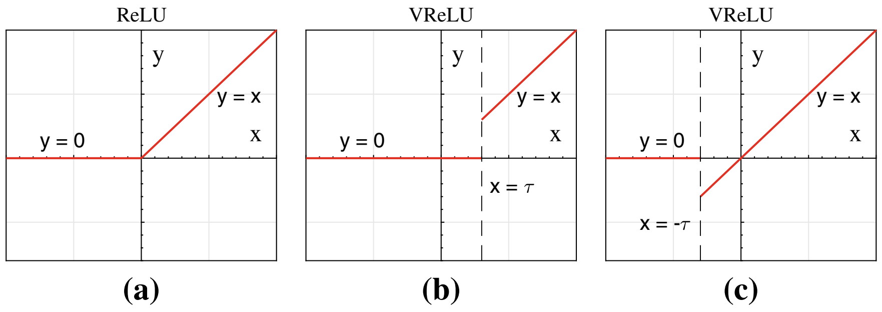


Fig. 6 Activation function in **a** Rectified Linear Units (ReLU), **b** Variable ReLU (VReLU) with positive threshold τ, and **c** VReLU with negative τ

$$f(x) = \begin{cases} x & x > \tau \\ 0 & x \leq \tau \end{cases}, \quad \tau \in N(0, 1) \tag{10}$$

where the fixed threshold in ReLU is replaced by a variable threshold τ which is randomly sampled from a normal distribution $N(0, 1)$ for each hidden unit at each epoch.

There is an illustration of VReLU in Fig. 6 for positive and negative values of τ and their comparison with the conventional ReLU. As shown in [28], this variable threshold augments the generalization of URBM model by modifying the histograms of GMM-RBM vectors. It has recently shown that VReLU is also effective for supervised speaker embedding extraction [91]. Experiments on the core test-common condition 5 of the NIST 2010 Speaker Recognition Evaluation (SRE) showed that GMM-RBM vectors could perform competitive to that of conventional i-vectors with both cosine and PLDA scoring but with much less computational load in the vector extraction stage, which is important for real-time applications. Moreover, fusing the scores of GMM-RBM vectors with i-vectors outperforms i-vectors.

Another efficient unsupervised speaker embedding is proposed by the authors in [29] using RBMs, where the speaker spectral features with a temporal context of $\{t-2, t-1, t, t+1, t+2\}$ are mapped into a single fixed-dimensional vector conveying speaker-specific information. This framework includes two stages, namely URBM training based on background data, and model adaptation for all the utterances. The URBM is considered as the speaker-independent model, which would lead to speaker-dependent models via model adaptation. The adaptation is carried out by training an RBM model, which is initialized by the URBM parameters. Data samples of the corresponding utterance are used for the adaptation procedure, however unlike URBM training, it is performed with just a few number of iterations. Unlike all the previous methods, in this framework the network parameters, connection weights, are utilized rather than using the network activations. In other words, the connection weights of each speaker-specific RBM model are concatenated to

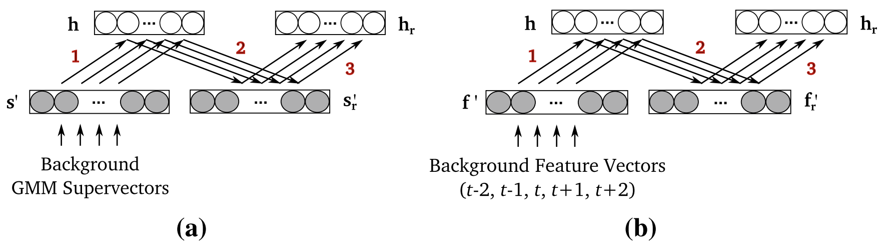

h
h$_r$
1
2
3
s'
s$_r'$
Background GMM Supervectors
(a)
h
h$_r$
f '
f$_r'$
Background Feature Vectors (t-2, t-1, t, t+1, t+2)
(b)

Fig. 7 Universal RBM (URBM) training for **a** GMM-RBM vector extraction, **b** RBM vector extraction

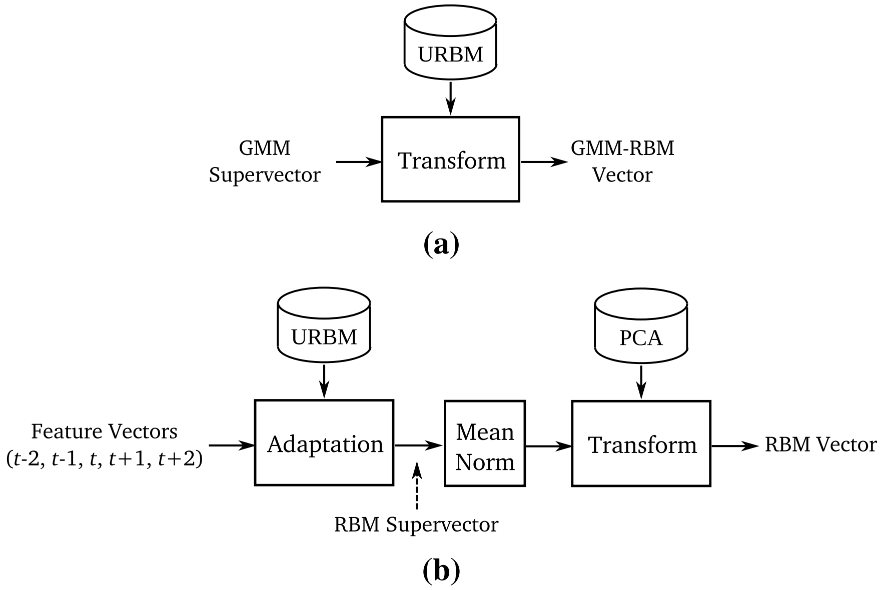


Fig. 8 Unsupervised speaker embedding extraction **a** GMM-RBM vector [28], **b** RBM vector [29]

build an RBM supervector. The dimension of these supervectors are later reduced using a PCA-whitening transformation and is referred to as RBM vectors.

Similar to i-vectors, these RBM embeddings can be used in SV using cosine or PLDA similarity. Experimental results on NIST SRE 2006 database show that RBM vectors achieve 15% and 24% relative improvements, in terms of EER, compared to i-vectors using cosine scoring when they are used alone and in combination with i-vectors, respectively. Figures 7 and 8 compare, respectively, URBM training and embedding extraction process for GMM-RBM and RBM vectors. For RBM vector extraction, a temporal context is considered and the computational cost is relatively higher compared to GMM-RBM vector extraction.

4 Deep Learning Backends

In this section, we describe how DL technology can be used as a backend on i-vectors. DL has been used on i-vectors for noise and reverberant effect compensation [101–104], domain adaptation [105], the compensation of the speech duration mismatch between train and test i-vectors [106, 107], and as an alternative scoring and session compensation technique to PLDA (e.g., [33, 35, 43, 96]).

As it was mentioned in Sect. 2.3, PLDA is one of the most effective backend techniques for i-vectors. It performs the scoring together with the compensation of the session variability, at the expense of speaker-labeled background data. Several DL-based techniques have been proposed to compete with PLDA. Most of these approaches use the speaker labels of the background data for training, as in PLDA, but mostly with no significant gain compared to PLDA. For example, RBMs have been proposed in [96] to learn speaker and channel factor sub-spaces in a PLDA training. A tied variational autoencoder architecture is proposed in [35] to simulate a nonlinear PLDA. A combination of RBM, autoencoder, and PLDA is proposed in [36, 37] for speaker and channel variability compensation, which shows some improvements compared to using only PLDA.

The story will be different once no speaker-labeled background data is available or only a few labeled data is accessible, which is the case in most real world applications. In this case, the powerful PLDA will not be applicable. Recently, a SR challenge was organized by NIST [38] to address how a comparable performance with PLDA can be achieved when the background data is not labeled. A possible solution for this scenario is to take advantage of unsupervised methods to automatically label the background speakers as proposed for instance in [39, 40]. However, these methods are not able to correctly estimate all the labels. Although the PLDA trained by estimated labels performs reasonably well on the given data [39, 40], the performance is still not comparable with Oracle PLDA [64].

In this section, we mainly focus on the work proposed by the authors in [43]. The goal is to decrease the performance gap between the unlabeled-based (cosine) and labeled-based (PLDA) scoring baseline systems when the background data is not labeled. The authors take advantage of the large amount of unlabeled background data and the unsupervised learning of DBNs to train a global model referred to as Universal DBN (UDBN). UDBN is then adapted to the little amount of data of each target speaker and used as a speaker-specific initial point for training a DNN to discriminate target and non-target input i-vectors. An impostor/non-target i-vector selection algorithm is proposed as well to keep the training process almost balanced. In summary, each target speaker is trained by a two-class DBN-DNN with speaker-dependent initialization (Fig. 9). Figure 10 shows how the connection weights, between the first two layers, of the UDBN are adapted to two different speakers.

The new challenging database NIST 2014 i-vector challenge [38] is used for the experiments. The advantage of this database is that the i-vectors for the background, train, and test sets are provided by NIST and, therefore, the baseline systems will

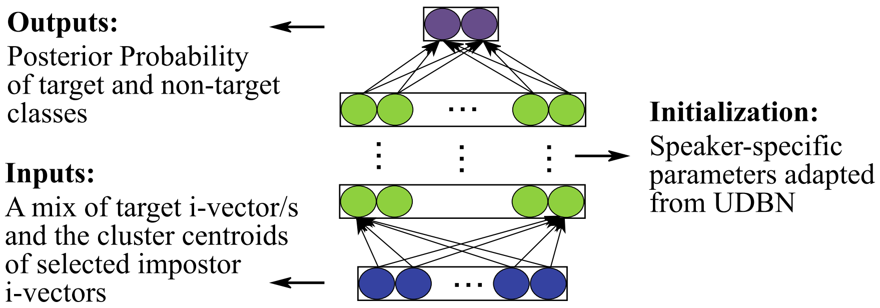

Outputs:
Posterior Probability of target and non-target classes
Inputs:
A mix of target i-vector/s and the cluster centroids of selected impostor i-vectors
Initialization:
Speaker-specific parameters adapted from UDBN

Fig. 9 A hybrid DBN-DNN backend architecture proposed to model each target speaker given target and non-target i-vectors [43]

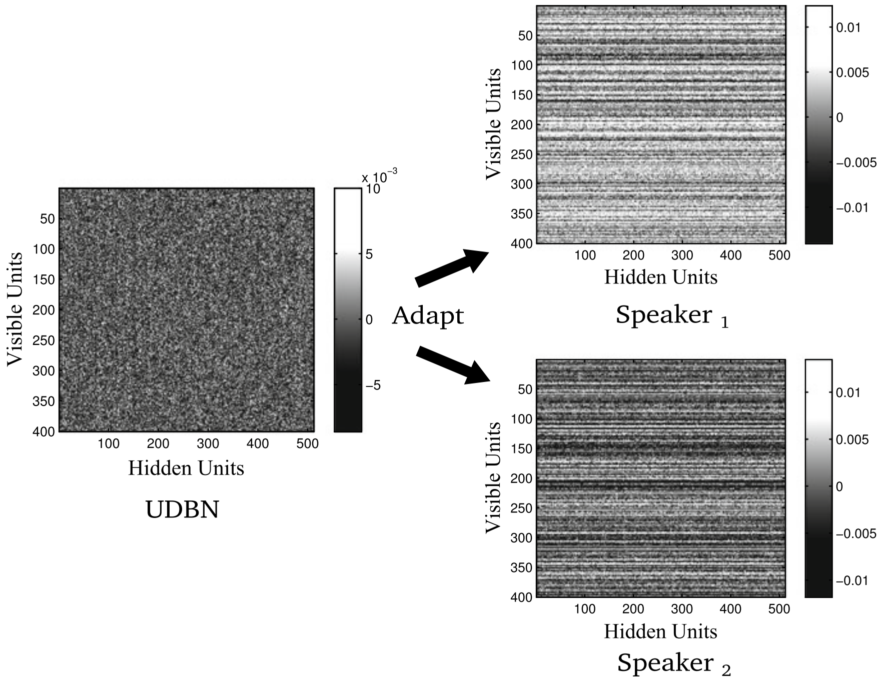


Fig. 10 Adaptation of the UDBN connection weights, between the first two layers, to two different speakers [43]

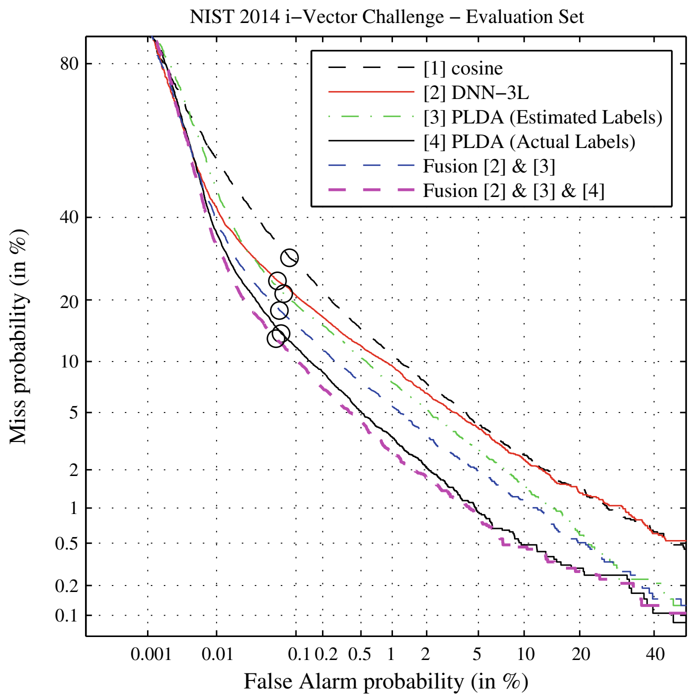


Fig. 11 Performance comparison of the proposed DL-based backend (Fig. 9) [43] with other commonly in use backends on the evaluation set of NIST 2014 i-vector challenge

have the same performance for everyone to compare new proposed systems with. Three baseline classification techniques are considered: cosine, PLDA with estimated labels, and PLDA with actual labels, which is called also Oracle PLDA system. Experimental results based on EER and minDCF (Table 1) and Detection Error Tradeo (DET) curves (Fig. 11) show that the proposed DL-based backend decreases the performance gap between cosine and Oracle PLDA scoring systems by 46% in terms of minimum DCF (minDCF) which is roughly similar to the PLDA scoring results obtained with unsupervised estimated labels. The interesting point is that the combination of the proposed backend and the PLDA with estimated labels in the score level is highly effective and decreases notably the performance gap by 79%. More comparison results and details can be found in [43].

5 Deep Learning End-to-Ends

An end-to-end SR system treats the entire training process, the frontend and the backend, as an integrated task. In other words, from the speaker spectral features all the way to the similarity scores, are trained jointly at once. The idea of an end-to-end

Table 1 Performance comparison of the proposed DL-based backend [43] with other commonly in use backends on NIST 2014 i-vector challenge dataset

	Progress set		Evaluation set	
	EER (%)	minDCF	EER (%)	minDCF
Unlabeled background data				
[1] cosine	4.78	0.386	4.46	0.378
[2] PLDA (Estimated Labels)	3.85	0.300	3.46	0.284
[3] Proposed DNN-1L	5.13	0.327	4.61	0.320
[4] Proposed DNN-3L	4.55	0.305	4.11	0.300
Fusion [2] & [4]	**2.99**	**0.260**	**2.70**	**0.243**
Labeled background data				
[5] PLDA (Actual Labels)	2.23	0.226	2.01	0.207
Fusion [2] & [5]	2.04	0.220	1.85	0.204
Fusion [4] & [5]	2.13	0.221	2.00	0.196
Fusion [2] & [4] & [5]	**1.88**	**0.204**	**1.74**	**0.190**

architecture for SR have been in use for a long time, e.g., in [2, 7]. This classification scheme was carried out mostly by learning a distinct classifying network for each individual speaker. In this sense for SV, the model of the claimed speaker to be verified, were fed by the feature vectors of the test utterance.

Recently, there have been several attempts to build an end-to-end SR system using DNNs. However, most of these efforts have addressed text-dependent SR as, e.g., in [108–110]. After successful attempts of applying UDBN and model adaptation as a backend to the i-vectors in [41], the same framework was proposed by the authors in [46] for an end-to-end system to discriminate between targets and impostors according to their speaker spectral features. However, the number of impostor samples, at the feature level, is significantly larger than the previous work which was based on i-vectors. Therefore, the impostor selection algorithms which were based on i-vectors were too costly to be feasible. Moreover, the number of target speaker samples varies from one speaker to another, making the training process more difficult. It was shown that the presented architecture in [46] was affected to a lesser extent by the problem of imbalance data training between impostor and enrollment utterances and was able to outperform the conventional end-to-end Multilayer Perceptron (MLP) models. However, they could not compete with the modern i-vector systems.

In another attempt, [111] proposed to operate directly on raw audio signals which are treated by a particular form of topology and weight evolving artificial neural networks [112] known as neuroevolution of augmenting topologies [113]. However, as the authors have stated in [111], it is just a proof-of-concept and the experiments present the potentials of the algorithm for further investigations.

6 Conclusions

Historically, SR has faced many challenges over the time. Demands for having higher accuracy, faster recognition, and more robustness against changing the environment have led to non-stop investigations as in other applications. Deep Learning (DL) is a new technology with which higher abstract features of the input data can be extracted. Hence, it has opened doors for new research in a wider range of signal processing and machine learning applications. DL has been used in all parts of a typical SR system from frontend to backend and even as a whole end-to-end system. As a frontend, it has been used for the extraction of the so-called bottleneck features, or as an acoustic model for the Baum-Welch statistic computation which is further used for i-vector extraction. Several network architectures and training processes have been proposed and substantial improvements are observed compared to classical acoustic models and features in the i-vector extraction process. However, the proposed DL architectures still suffer from some shortcomings. For instance, the DL acoustic models are usually language dependent, need phonetic labels, and increase the computational cost. On the other hand, deep network architectures have been used to extract directly a compact representation of the speaker characteristics of a speech utterance, which are often referred to as speaker embeddings. These low dimensional vectors are proposed as alternatives to conventional i-vectors. Recent investigations have shown that data augmentation can significantly improve the performance of speaker embeddings and make them competitive with traditional i-vectors not only for short segments but also for long-duration speech segments. The proposed DL techniques for backend have been mostly supervised, i.e., given the speaker labeled background data, and have tried to be alternatives to PLDA scoring. These techniques still do not show very competitive performance to PLDA when they are used as standalone systems, implying that there are still a lot of room for improvements. One of the concerns in SR is that the speaker labels are usually expensive to obtain and are not always accessible. Therefore, it would be difficult to use PLDA or other DL based alternative techniques. The challenge will, therefore, be how one technique can decrease the performance gap between unsupervised and supervised techniques when the background data is not labeled. This is one of the areas that has not been fully explored. Recently, some end-to-end systems have been proposed based on DL, meaning that both frontend and backends are performed at once in a single system. Although the reported results have been competitive to the conventional systems in some cases, still much more accuracy gain is expected. One of the important and promising observations on the use of DL in SR is that even if it has not been able to win the traditional signal processing techniques in most cases, it has often been effective in combination with them. Over the last few years, the authors have tried to be active in doing research in all parts of a SR system, specifically to investigate in more challenging areas like unsupervised speaker embeddings, unsupervised DL based backends, and to build end-to-end systems all based on DL architectures.

Acknowledgements This work is partially supported by the Spanish project DeepVoice under grant number TEC2015-69266-P.

References

1. Oglesby, J., Mason, J.S.: Speaker identification using neural nets. IOA Speech (1988)
2. Oglesby, J., Mason, J.S.: Speaker recognition with a neural classifier. In: Artificial Neural Networks, IET (1989)
3. Oglesby, J., Mason, J.S.: Optimisation of neural models for speaker identification. In: ICASSP (1990)
4. Bennani, Y., Soulie, F.F., Gallinari, P.: A connectionist approach for automatic speaker identification. In: ICASSP (1990)
5. Bennani, Y., Gallinari, P.: On the use of tdnn-extracted features information in talker identification. In: ICASSP (1991)
6. Oglesby, J., Mason, J.S.: Radial basis function networks for speaker recognition. In: ICASSP (1991)
7. Rudasi, L., Zahorian, S.A.: Text-independent talker identification with neural networks. In: ICASSP (1991)
8. Farrell, K.R., Mammone, R.J., Assaleh, K.T.: Speaker recognition using neural networks and conventional classifiers. IEEE Trans. Speech Audio Process. (1994)
9. Heck, L.P., Konig, Y., Sönmez, M.K., Weintraub, M.: Robustness to telephone handset distortion in speaker recognition by discriminative feature design. Speech Commun. (2000)
10. Yegnanarayana, B., Kishore, S.P.: Aann: an alternative to gmm for pattern recognition. Neural Netw. (2002)
11. Lapidot, I., Guterman, H., Cohen, A.: Unsupervised speaker recognition based on competition between self-organizing maps. IEEE Trans. Neural Netw. (2002)
12. Chen, K., Salman, A.: Learning speaker-specific characteristics with a deep neural architecture. IEEE Trans. Neural Netw. (2011)
13. Chen, K., Salman, A.: Extracting speaker-specific information with a regularized siamese deep network. In: Advances in Neural Information Processing Systems (2011)
14. Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., Ouellet, P.: Front-end factor analysis for speaker verification. Speech, and Language Processing, IEEE Transactions on Audio (2011)
15. Lei, Y., Scheffer, N., Ferre, L., Mclaren, M.: A novel scheme for speaker recognition using a phonetically-aware deep neural network. In: ICASSP (2014)
16. Kenny, P., Gupta, V., Stafylakis, T., Ouellet, P., Alam, J.: Deep neural networks for extracting baum-welch statistics for speaker recognition. In: Odyssey (2014)
17. Campbell, W.M.: Using deep belief networks for vector-based speaker recognition. In: Interspeech (2014)
18. Richardson, F., Reynolds, D., Dehak, N.: Deep neural network approaches to speaker and language recognition. IEEE Signal Process. Lett. (2015)
19. Garcia-Romero, D., Zhang, X., McCree, A., Povey, D.: Improving speaker recognition performance in the domain adaptation challenge using deep neural networks. In: SLT (2014)
20. Mclaren, M., Lei, Y., Ferre, L.: Advances in deep neural network approaches to speaker recognition. In: ICASSP (2015)
21. Variani, E., Lei, X., McDermott, E., Lopez Moreno, I., Gonzalez-Dominguez, J.: Deep neural networks for small footprint text-dependent speaker verification. In: ICASSP (2014)
22. Wang, S., Qian, Y., Yu, K.: What does the speaker embedding encode? In: Interspeech (2017)
23. Bhattacharya, G., Alam, J., Kenny, P.: Deep speaker embeddings for short-duration speaker verification. In: Interspeech (2017)
24. Snyder, D., Garcia-Romero, D., Povey, D., Khudanpur, S.: Deep neural network embeddings for text-independent speaker verification. In: Interspeech (2017)
25. Snyder, D., Garcia-Romero, D., Sell, G., D. Povey, Khudanpur, S.: X-vectors: robust dnn embeddings for speaker recognition. In: ICASSP (2018)
26. Prince, S.J.D., Elder, J.H.: Probabilistic linear discriminant analysis for inferences about identity. In: ICCV (2007)
27. Ghahabi, O., Hernando, J.: Restricted boltzmann machine supervectors for speaker recognition. In: ICASSP (2015)

28. Ghahabi, O., Hernando, J.: Restricted Boltzmann machines for vector representation of speech in speaker recognition. Comput. Speech Lang. (2018)
29. Safari, P., Ghahabi, O., Hernando, J.: From features to speaker vectors by means of restricted boltzmann machine adaptation. In: Odyssey (2016)
30. Kenny, P.: Bayesian speaker verification with heavy tailed priors. In: Odyssey (2010)
31. Stafylakis, T., Kenny, P., Senoussaoui, M., Dumouchel, P.: Preliminary investigation of boltzmann machine classifiers for speaker recognition. In: Odyssey (2012)
32. Senoussaoui, M., Dehak, N., Kenny, P., Dehak, R., Dumouchel, P.: First attempt of boltzmann machines for speaker verification. In: Odyssey (2012)
33. Novoselov, S., Pekhovsky, T., Simonchik, K., Shulipa, A.: RBM-PLDA subsystem for the NIST i-vector challenge. In: Interspeech (2014)
34. Isik, Y.Z., Erdogan, H., Sarikaya, R.: S-vector: a discriminative representation derived from i-vector for speaker verification. In: EUSIPCO (2015)
35. Villalba, J., Brümmer, N., Dehak, N.: Tied variational autoencoder backends for i-vector speaker recognition. In: Interspeech (2017)
36. Novoselov, S., Pekhovsky, T., Kudashev, O., Mendelev, V.S., Prudnikov, A.: Non-linear plda for i-vector speaker verification. In: Interspeech (2015)
37. Pekhovsky, T., Novoselov, S., Sholokhov, A., Kudashev, O.: On autoencoders in the i-vector space for speaker recognition. In: Odyssey (2016)
38. The NIST Speaker Recognition i-vector Machine Learning Challenge (2014)
39. Khoury, E., El Shafey, L., Ferras, M., Marcel, S.: Hierarchical speaker clustering methods for the nist i-vector challenge, In: Odyssey (2014)
40. Novoselov, S., Pekhovsky, T., Simonchik, K.: STC speaker recognition system for the NIST i-vector challenge. In: Odyssey (2014)
41. Ghahabi, O., Hernando, J.: Deep belief networks for i-vector based speaker recognition. In: ICASSP (2014)
42. Ghahabi, O., Hernando, J.: i-vector modeling with deep belief networks for multi-session speaker recognition. In: Odyssey (2014)
43. Ghahabi, O., Hernando, J.: Deep learning backend for single and multisession i-vector speaker recognition. Speech, and Language Processing, IEEE/ACM Transactions on Audio (2017)
44. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 (2014)
45. Song, W., Cai, J.: End-to-end deep neural network for automatic speech recognition (2015)
46. Safari, P., Ghahabi, O., Hernando, J.: Feature classification by means of deep belief networks for speaker recognition. In: EUSIPCO (2015)
47. Reynolds, D.A., Rose, R.C.: Robust text-independent speaker identification using gaussian mixture speaker models. IEEE Trans. Speech Audio Process. (1995)
48. Campbell, W.M., Campbell, J.P., Reynolds, D.A., Singer, E., Torres-Carrasquillo, P.A.: Support vector machines for speaker and language recognition. Comput. Speech Lang. (2006)
49. Sadaoki, F.: Fifty years of progress in speech and speaker recognition. J. Acoust. Soc. Am. (2004)
50. Nadeu, C., Hernando, J., Gorricho, M.: On the decorrelation of filter-bank energies in speech recognition. In: Eurospeech (1995)
51. Nadeu, C., Macho, D., Hernando, J.: Time and frequency filtering of filter-bank energies for robust HMM speech recognition. Speech Commun. (2001)
52. Hansen, J.H.L., Hasan, T.: Speaker recognition by machines and humans: a tutorial review. IEEE Signal Process. Mag. (2015)
53. Reynolds, D.A., Quatieri, T.F., Dunn, R.B.: Speaker verification using adapted gaussian mixture models. In: Digital Signal Processing (2000)
54. Campbell, W.M., Sturim, D.E., Reynolds, D.A.: Support vector machines using GMM supervectors for speaker verification. IEEE Signal Process. Lett. (2006)
55. Dehak, N., Chollet, G.: Support vector gmms for speaker verification. In: Odyssey (2006)
56. Lee, K., You, C., Li, H., Kinnunen, T., Zhu, D.: Characterizing speech utterances for speaker verification with sequence kernel SVM. Comput. Speech Lang. (2008)

57. Campbell, W.M., Sturim, D.E., Reynolds, D.A., Solomonoff, A.: SVM based speaker verification using a GMM supervector kernel and NAP variability compensation. In: ICASSP (2006)
58. Solomonoff, A., Campbell, W.M., Boardman, I.: Advances in channel compensation for SVM speaker recognition. In: ICASSP (2005)
59. Hatch, A.O., Stolcke, A.: Generalized linear kernels for one-versus-all classification: application to speaker recognition. In: ICASSP (2006)
60. Kenny, P.: Joint factor analysis of speaker and session variability: theory and algorithms (2006)
61. Dehak, N., Dehak, R., Glass, J., Reynolds, D., Kenny, P.: Cosine similarity scoring without score normalization techniques. In: Odyssey (2010)
62. Garcia-Romero, D., Espy-Wilson, C. Y.: Analysis of i-vector length normalization in speaker recognition systems. In: Interspeech (2011)
63. Matějka, P., Glembek, O., Castaldo, F., Alam, J., Plchot, O., Kenny, P., Burget, L., Černocky, J.: Full-covariance ubm and heavy-tailed plda in i-vector speaker verification. In: ICASSP (2011)
64. Greenberg, C., Banse, D., Doddington, G., Garcia-Romero, D., Godfrey, J., Kinnunen, T., Martin, A., McCree, A., Przybocki, M., Reynolds, D.: The NIST 2014 speaker recognition i-vector machine learning challenge. In: Odyssey
65. Kinnunen, T., Li, H.: An overview of text-independent speaker recognition: from features to supervectors. Speech Commun. (2010)
66. Hansen, J.H.L., Hasan, T.: Speaker recognition by machines and humans: a tutorial review. IEEE Signal Process. Mag. (2015)
67. Ghahabi, O.: Deep learning for i-vector speaker and language recognition. Ph.D. thesis, Universitat Politècnica de Catalunya (2018)
68. Ferrer, L., Lei, Y., McLaren, M., Scheffer, N.: Spoken language recognition based on senone posteriors. In: INTERSPEECH (2014)
69. Ferrer, L., Lei, Y., McLaren, M., Scheffer, N.: Study of senone-based deep neural network approaches for spoken language recognition. IEEE/ACM Trans. Audio Speech Lang. Process. (2016)
70. Silnova, A., Burget, L., Cernocky, J.: Alternative approaches to neural network based speaker verification. In: Interspeech (2017)
71. Ranjan, S., Hansen, J.H.L.: Improved gender independent speaker recognition using convolutional neural network based bottleneck features. In: Interspeech (2017)
72. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (2014)
73. Serdyuk, D., Audhkhasi, K., Brakel, P., Ramabhadran, B., Thomas, S., Bengio, Y.: Invariant representations for noisy speech recognition. arXiv:1612.01928 (2016)
74. Shinohara, Y.: Adversarial multi-task learning of deep neural networks for robust speech recognition. In: INTERSPEECH (2016)
75. Yu, H., Tan, Z.-H., Ma, Z., Guo, J.: Adversarial network bottleneck features for noise robust speaker verification. arXiv:1706.03397 (2017)
76. Li, L., Tang, Z., Wang, D., Zheng, T.F.: Full-info training for deep speaker feature learning. In: ICASSP (2018)
77. Novoselov, S., Shulipa, A., Kremnev, I., Kozlov, A., Shchemelinin, V.: On deep speaker embeddings for text-independent speaker recognition. In: Odyssey (2018)
78. Li, L., Tang, Z., Shi, Y., Wang, D.: Gaussian-constrained training for speaker verification. arXiv:1811.03258 (2018)
79. Zeinali, H., Burget, L., Rohdin, J., Stafylakis, T., Cernocky, J.: How to improve your speaker embeddings extractor in generic toolkits. arXiv:1811.02066 (2018)
80. Huang, Z., Wang, S., Yu, K.: Angular softmax for short-duration text-independent speaker verification. In: Interspeech (2018)
81. Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., Wu, Y.: Learning fine-grained image similarity with deep ranking. In: CVPR (2014)

82. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: International Workshop on Similarity-Based Pattern Recognition. Springer (2015)
83. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: CVPR (2015)
84. Zhang, C., Koishida, K.: End-to-end text-independent speaker verification with triplet loss on short utterances. In: Interspeech (2017)
85. Bredin, H.: Tristounet: triplet loss for speaker turn embedding. In: ICASSP (2017)
86. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
87. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
88. Nagrani, A., Chung, J.S., Zisserman, A.: Voxceleb: a large-scale speaker identification dataset. In: Interspeech (2017)
89. Chung, J.S., Nagrani, A., Zisserman, A.: Voxceleb2: deep speaker recognition. In: Interspeech (2018)
90. India, M., Safari, P., Hernando, J.: Self multi-head attention for speaker recognition. In: Interspeech (2019)
91. Ghahabi, O., Fischer, V.: Speaker-corrupted embeddings for online speaker diarization. In: Interspeech (2019)
92. Jung, J.-W., Heo, H.-S., Yang, I.-H., Shim, H.-J., Yu, H.-J.: Avoiding speaker overfitting in end-to-end dnns using raw waveform for text-independent speaker verification (2018)
93. Ravanelli, M., Bengio, Y.: Speaker recognition from raw waveform with sincnet. arXiv:1808.00158 (2018)
94. Stafylakis, T., Kenny, P.: Preliminary investigation of boltzmann machine classifiers for speaker recognition. In: Odyssey (2012)
95. Senoussaoui, M., Dehak, N., Kenny, P., Dehak, R.: First attempt of boltzmann machines for speaker verification. In: Odyssey (2012)
96. Stafylakis, T., Kenny, P., Senoussaoui, M., Dumouchel, P.: PLDA using gaussian restricted boltzmann machines with application to speaker verification. In: Interspeech (2012)
97. Lee, H., Pham, P., Ng, A.Y.: Unsupervised feature learning for audio classification using convolutional deep belief networks. In: Advances in Neural Information Processing Systems (2009)
98. Ghahabi, O., Hernando, J.: Global impostor selection for DBNs in multi-session i-vector speaker recognition. In: Advances in Speech and Language Technologies for Iberian Languages. Springer International Publishing (2014)
99. Campbell, W.M.: Using deep belief networks for vector-based speaker recognition. In: Interspeech (2014)
100. Vasilakakis, V., Cumani, S., Laface, P.: Speaker recognition by means of deep belief networks. In: Biometric Technologies in Forensic Science (2013)
101. Mahto, S., Yamamoto, H., Koshinaka, T.: I-vector transformation using a novel discriminative denoising autoencoder for noise-robust speaker recognition. In: Interspeech (2017)
102. Alam, J., Kenny, P., Bhattacharya, G., Kockmann, M.: Speaker verification under adverse conditions using i-vector adaptation and neural networks. In: Interspeech (2017)
103. Guzewich, P., Zahorian, S.: Improving speaker verification for reverberant conditions with deep neural network dereverberation processing. in: Interspeech (2017)
104. Tan, Z., Mak, M.-W.: I-vector dnn scoring and calibration for noise robust speaker verification. In: Interspeech (2017)
105. Shon, S., Mun, S., Kim, W., Ko, H.: Autoencoder based domain adaptation for speaker recognition under insufficient channel information. arXiv:1708.01227 (2017)
106. Bousquet, P.-M., Rouvier, M.: Duration mismatch compensation using four-covariance model and deep neural network for speaker verification. In: Interspeech (2017)
107. Guo, J., Nookala, U. A., Alwan, A.: Cnn-based joint mapping of short and long utterance i-vectors for speaker verification using short utterances. In: Interspeech (2017)

108. Heigold, G., Moreno, I., Bengio, S., Shazeer, N.: End-to-end text-dependent speaker verification. In: ICASSP (2016)
109. Zhang, S.-X., Chen, Z., Zhao, Y., Li, J., Gong, Y.: End-to-end attention based text-dependent speaker verification. In: SLT (2016)
110. Heo, H.-S., Jung, J.-W., Yang, I.-H., Yoon, S.-H., Yu, H.-J.: Joint training of expanded end-to-end dnn for text-dependent speaker verification. In: Interspeech (2017)
111. Valenti, G., Daniel, A., Evans, N.: End-to-end automatic speaker verification with evolving recurrent neural networks. In: Odyssey (2018)
112. Dasgupta, D., McGregor, D.R.: Designing application-specific neural networks using the structured genetic algorithm. In: COGANN (1992)
113. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolut. Comput. (2002)

Baby Cry Detection: Deep Learning and Classical Approaches

Rami Cohen, Dima Ruinskiy, Janis Zickfeld, Hans IJzerman and Yizhar Lavner

Abstract In this chapter, we compare deep learning and classical approaches for detection of baby cry sounds in various domestic environments under challenging signal-to-noise ratio conditions. Automatic cry detection has applications in commercial products (such as baby remote monitors) as well as in medical and psychosocial research. We design and evaluate several convolutional neural network (CNN) architectures for baby cry detection, and compare their performance to that of classical machine-learning approaches, such as logistic regression and support vector machines. In addition to feed-forward CNNs, we analyze the performance of recurrent neural network (RNN) architectures, which are able to capture temporal behavior of acoustic events. We show that by carefully designing CNN architectures with specialized non-symmetric kernels, better results are obtained compared to common CNN architectures.

Keywords Baby cry detection · Deep learning · Convolutional neural networks · Audio detection

R. Cohen (✉)
Viterbi Faculty of Electrical Engineering, Technion - Israel Institute of Technology, Haifa, Israel
e-mail: rc@technion.ac.il

D. Ruinskiy · Y. Lavner
Department of Computer Science, Tel-Hai College, Upper Galilee, Israel
e-mail: dima.ruinskiy@gmail.com

Y. Lavner
e-mail: yizhar.lavner@gmail.com

D. Ruinskiy
Network.IO Innovation Lab, Haifa, Israel

J. Zickfeld
Department of Psychology, University of Oslo, Oslo, Norway
e-mail: jhzickfeld@gmail.com

H. IJzerman
LIP/PC2S, Université Grenoble Alpes, Grenoble, France
e-mail: h.ijzerman@gmail.com

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_7

1 Introduction

In recent years, *deep neural networks* have been used with great success in a variety of practical real-world problems. As opposed to the traditional feature-extraction stage in classical machine-learning algorithms, deep neural networks automate the formation of useful features from the data. Such networks consist of multiple layers, resembling the way computations are performed in the brain. In these layers, a hierarchy of non-linear features is formed, growing in complexity with the depth of the network. A combination of these features is used in the last layer of the network to generate a prediction.

The most impressive results are perhaps in the area of computer vision, starting with the seminal work of [1], where deep learning networks exhibit state-of-the-art performance in various tasks, such as object detection and classification. With the advances in deep-learning techniques and the availability of large databases for training, deep learning is also becoming an important tool for automatic audio event detection [2, 3].

In this chapter, we compare deep-learning and classical approaches for the detection of baby cry events in acoustic signals. Accurate and reliable detection of infant cry events in a stream of audio is a prerequisite for classification algorithms and screening tasks, which rely on the acoustic properties of the cry. One of the main difficulties in detecting baby cry in a domestic environment or in other natural environments, such as neonatal clinic units or nurseries, is the presence of noise and background sounds–speech, music, electronic toys, door opening, phone ringing, and many others. This poses a considerable challenge for classical machine-learning approaches, which typically start by extracting a set of distinguishing features from the acoustic signal. Background noise may have fundamental frequency or vocal qualities similar to those of infant cry, hindering the detection algorithm. In addition, the signal-to-noise ratio (SNR) often varies. Speech in particular poses a considerable challenge for the detection, due to frequency content similar to baby cry, which may introduce false-positive events.

In the research described here we devise and evaluate deep-learning approaches for baby cry detection. We design specialized convolutional neural networks (CNNs) for this task and study appropriate image representations of audio signals for serving as inputs to the CNNs. We use the non-linear log Mel-filter bank (log MFB) representation, where each pixel represents a frequency range according to the logarithmic Mel-scale; this representation is known to capture well the relevant frequencies that distinguish different types of the acoustic signals. We compare the performance of our CNN architectures to traditional machine-learning algorithms, such as logistic regression and support vector machine (SVM) classifiers.

The performance evaluation is carried out using an annotated database containing several hours of recordings of babies in domestic environments. In addition to baby cry, these recordings contain various types of domestic sounds, such as phone ringing, door opening and parent speech. We discuss the trade-off between the false-positive rate and the detection rate, based on a receiver operating characteristic (ROC) curve, and provide performance analysis of CNN detection results with a varying number of layers and units. We show considerable performance gain compared to classical machine-learning approaches, especially at the low false-positive rate regime.

1.1 Approaches in Audio Event Detection

Audio event detection is the task of spotting specific acoustic events within long clips or streams of audio data. Examples of acoustic events are human voices, specific utterances, different types of domestic or urban noises, sounds produced by various musical instruments, and many others. The detection task can have varying requirements: from simple spotting (presence or absence of the event in question, such as in a voice activity detector [4]) to accurate demarcation of the event boundaries (onset and offset), e.g., [5]; in some cases detection of multiple event types may be desired, which introduces a related task of sound event *classification*: assigning each audio portion to one of several pre-defined classes (sometimes referred to as *annotation*). The classification may be applied only to the detected events, for example distinguishing different types of fricative consonants [6, 7], or to entire segments, for instance, when discriminating between speech and music [8, 9], or between various musical genres [10].

Automatic detection and classification of acoustic events in audio signals is a major research field in machine learning, due to its many applications. In the broadest sense, it is a key part in *auditory machine perception* (also known as *machine listening*, [11])—where a computer learns to interpret and analyze audio information similarly to a human. Some examples of specific applications are speech recognition [12], voice-controlled appliances [13], audio surveillance [14], health monitoring [15] and audio signal enhancement [5, 16].

One of the early implementations of deep learning for acoustic event detection [17], used a convolutional neural network (CNN) for a robust sound event recognizer in different kinds of noisy environments. In [18], a deep model consisting of 2 convolutional layers with max-pooling and 2 fully connected layers was used for detecting various urban and environmental sounds. A CNN architecture was also employed for acoustic scene classification task in [19]. Reference [20] showed that a convolutional recurrent neural network (RNN) approach yielded better results in polyphonic sound detection (where multiple types of events can be detected simultaneously) than separate CNN/RNN or traditional approaches. A similar approach was used in [21] for a low-latency monaural sound source separation system. Applicability of deep learning approaches for automatic music tagging (assigning properties such as genre, instrumentation, rhythm, etc.) was also demonstrated [22, 23].

Other types of deep learning architectures have also been investigated. For example, in [24], the authors used a deep network consisting of convolutional layers, followed by fully connected layers and a single-element output layer with a sigmoid activation function; [25] used a topology known as Capsule Network (CapsNet, [26]) and found that it could outperform a traditional CNN. Another study compared deep learning methods to a hand-crafted support vector data description (SVDD) classifier with a few carefully selected features, and found that the latter can achieve comparable performance to a CNN at lower computational cost, but with the drawback of having to design features specific to the task [27].

Various methods have also been investigated for the specific task of cry detection; a good survey of traditional approaches is available in [28]. In recent years, advances in deep learning made it a popular technique as well. In [29], a specially-designed CNN was shown to outperform a traditional logistic regression-based classifier in very low false-positive rate regimes. In [30], a CNN running on audio captured from a microphone array installed next to a baby carriage could detect cry with 86% accuracy. Reference [31] compared a CNN followed by a Hidden Markov Model (HMM) to a Linear Discriminant Analysis (LDA) classifier.

An automatic baby cry detector has many applications: it is commonly employed in safety-related devices, such as baby monitors [32], and has been proposed as part of a system to detect children forgotten in vehicles [33]; some commercial products featuring cry detection technology include [34–36]. Identification, followed by classification of the cry signals, can be useful for medical purposes, such as detection of pathologies based on the acoustic properties of the cry signal (e.g., [37, 38]), or assessment of the neurological state of infants based on differences in the crying between full-term and preterm babies [39].

1.2 The Origin and Role of Infant Cry

The human cry has several roles, depending on developmental stage. The initial function, and thus developmental origin, is relatively clear. For some of the other aspects, there are generally-accepted theories and hypotheses, but not solid conclusions, as we are not sufficiently confident of psychological theory as a means for prediction. The vocal cry is one of the first forms of communication to interact with the caregiver [40], and is common to more species than humans. Reference [41] proposed that feedback-sensitive attachment behavior is vital to retain the caregiver's proximity, and that crying is one of the most important channels for establishing it [42–44]. In humans, the infant depends on the caregiver for food, safety and warmth [45]. As a result, much of our functioning focuses on a kind of "co-regulation" that caregivers provide to infants early in life, and adult partners to each other later in life [46].

Crying in human infants is elicited from rhythmical transitions between inhalation and exhalation, due to a vibration of the vocal cords that produces periodic air pulses. The period of these pulses is called the fundamental frequency (pitch), and its typical values in healthy babies are $250-600$ Hz. The cry signal is shaped by the vocal tract, leading to resonant frequencies termed as formants. The first two formants occur typically around 1100 Hz and 3300 Hz, respectively [47]. Some studies [44, 48] point out that humans early in life already display various forms of crying: protest crying (in which the infant faces loss, like being left in the crib, and wants to undo the loss), sad crying of despair (a low wail signifying acceptance of loss), and detached inhibited crying (typically an absence of outward crying, associated with a life-threatening separation from the caregiver). Sometimes additional types, such as hunger and pain crying are considered.

For a long time, researchers assumed that different types of cries cannot be reliably distinguished, as even mothers are not always accurate at discriminating pain versus hunger crying (e.g., [43]). Nowadays, with drastically improved accuracy of measurement equipment and analysis software, and with much larger sample sets available, there are reasons to doubt this assumption. Recent research suggests that one can reliably distinguish between pain, hunger, sadness, fear, and anger cries on the basis of facial expressions and cry characteristics [49, 50], and it seems plausible that deep learning methods can be applied for successful automatic classification of different types of cry.

The type, frequency, and duration of crying are highly variable, especially after early infancy (e.g., [43, 51]). Crying peaks at about 6 weeks and then declines until 4 months after which it remains rather stable [52]. The development of crying depends in part on the caregiver's response to the cry (or the lack of it). In young infants, if the caregiver responds (by holding, touching, and/or feeding) crying typically ceases [53]. A positive response from the caregiver signals that the infant can rely on others to help meet environmental demands, while a lack of it signals that the infant needs to cope with environmental demands itself. Later in life, the type, frequency, and duration may depend on an individual's temperament [54, 55], or attachment [56]. For example, infants whose mothers did not respond consistently to their cry later started oscillating between clinging to their mother and resisting contact (see e.g., [43]). In many cases, infants adapt to whatever is required for survival (for example, self-reliance in case of a non-responsive caregiver); however, differences in crying related to such adaptations have not yet been fully investigated.

There is no complete knowledge of the characteristics of pathological cry, in part due to a lack of sufficiently large samples and accurate methodologies. At the very least, a cry signal with fundamental frequency (pitch) above 600 Hz has been generally regarded as indicating health issues [57]. Another function of crying may thus well be to convey the infant's fitness, preventing caregivers from investing [47]. Pain crying usually starts abruptly, is usually longer, and often involves hyperphonated cries (characterized by high pitch; [43]). A commonly accepted pathology, known as 'colic', is indicated by excessive crying; researchers typically distinguish between intrinsic causes for colic, such as food allergies, or extrinsic ones, for example inappropriate physical contact.

Classification of normal and pathological infant crying (especially in newborns) has been the subject of extensive research. Some studies focused on a specific pathology, such as deafness, or respiratory distress syndrome (RSD), while in others multiple pathologies were investigated [58]. Differences of cry between normal babies and those with high-risk for autism have been studied as well [59, 60]. A recent survey of the research status is available [61].

While pathological cry in newborns has been extensively studied, little is known about how crying early in life relates to potential maladaptive behaviors later in life. Large samples of crying recordings, combined with novel machine learning and deep learning techniques can help identify possible connections (for examples in psychology, see [62, 63]). One of the difficulties with research in the early stages of life is that it is often too intrusive. To make it less intrusive, we have integrated automatic cry detection in an Android-based smartphone app.[1] This smartphone app can be used to detect, in real-time, baby cry events, in the infant's earliest days. Usage of the app allows the infant cry to be correlated to other variables (like caregiver peripheral temperature).

2 Deep Learning Approach

Convolutional neural networks (CNNs) [64, 65] have wide applications in the fields of computer vision, natural language processing and many others, especially where huge amounts of data have to be processed and classified. Like ordinary neural networks, CNNs consist of multiple layers connected by neurons that have learnable weights. Commonly used layer types are: convolutional layers (applying convolution / dot product operation), pooling layers (combining outputs of several neurons into a single neuron in the next layer), rectified linear unit (ReLU) layers, fully-connected (dense) layers and more. The exact number and configuration of layers is application-dependent.

CNNs learn the parameters (usually termed as *weights*) of each layer in a training process. This process is carried out using a gradient descent approach and the backpropagation technique [66]. The complexity of the training process scales with the number of *trainable* parameters of the network. In typical CNNs, there might be thousands to millions of parameters, whose values are learned in the training stage.

In contrast to traditional classification algorithms (e.g., support vector machines), no features are typically extracted from the data prior to CNN-based classification and detection. Instead, the input to a CNN usually consists of the raw data, e.g. images, which is one of their primary advantages. However, the performance of CNN-based classifiers depends heavily on the architectural structure of the CNN in use. Designing a CNN architecture requires choosing the number of layers, the kernel size, connectivity between layers and more.

In this section, we design and evaluate multiple CNN architectures for the detection of baby cry, operating on log Mel-filter bank representation of the audio data.

2.1 Data Representation

In image classification or detection tasks based on CNNs, the input to the network is typically composed of the raw images. The CNN role is then to efficiently extract

[1]Code available at https://github.com/co-relab/bioapp.

spatial features from the input images and to propagate them to deeper layers, such that correct prediction is obtained at the CNN output. For example, the detection of an object such as a cat can be viewed as the detection of its eyes, mouth and tail, by dedicated filters. However, using raw audio signals as an input to a CNN is typically undesired, as the convolution filters will be applied to temporally-adjacent samples. When dealing with sampling rates such as 44,100 Hz, the output of such filters is of limited benefit, in particular when typical small (one-dimensional) kernels are used.

To better exploit the power of CNNs for audio classification tasks, it is often beneficial to convert audio signals to an image representation with meaningful spatial information. A natural approach for this aim is the use of time-frequency representations. The most common type of such representation is spectrograms, generated by applying the short-time Fourier transform to the data. However, the linear scale of the spectrogram (both in time and frequency domains) makes it difficult to separate simultaneous sounds with similar frequency content based. Thus, the efficiency of spectrogram representation for our task is likely to be of limited benefit, in particular in presence of noise with characteristics similar to those of cry signals.

To improve the robustness of the CNNs to noise, we use a *log Mel-filter bank* (log MFB, LMFB) representation [67] of the audio signals. The Mel-scale aims to mimic the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. This logarithmic representation is often beneficial for sound classification, as it better separates different types of signals with similar frequency content. The main difference between MFB and Mel-Frequency Cepstrum coefficients (MFCC) [67] is that the discrete cosine transform (DCT) of the log-power spectrum is skipped in MFB. This is because DCT decorrelates the data, whereas spatial correlation of the input is actually advantageous for a CNN. The reason is that the 2D convolution operation used in convolutional layers is motivated by the correlation between neighbouring pixels in natural images.

To produce a log MFB representation of the data, the input audio signal is divided into consecutive segments of 4096 samples each. These segments are further divided into frames of 512 samples each, with a step size of 128 samples. As the contribution of high frequency bands to the detection of cry signals is limited, a low-pass filter at 11,025 Hz is applied to the signal. A log MFB representation is then produced for each frame, using 50 triangular filters distributed according to the Mel scale in the frequency range [0, 11025] Hz. Given segments of 4096 samples and a step size of 128 samples, this leads to a 50×29 “image” representation of each segment. Another representation which has been used is the log Linear Filter Bank (LFB), which is produced using the same procedure as the MFB but with linearly-distributed filters. An example is shown in Fig. 1. In this figure, a short segment of a baby cry signal waveform is shown, with the corresponding spectrogram, MFB and LFB representations.

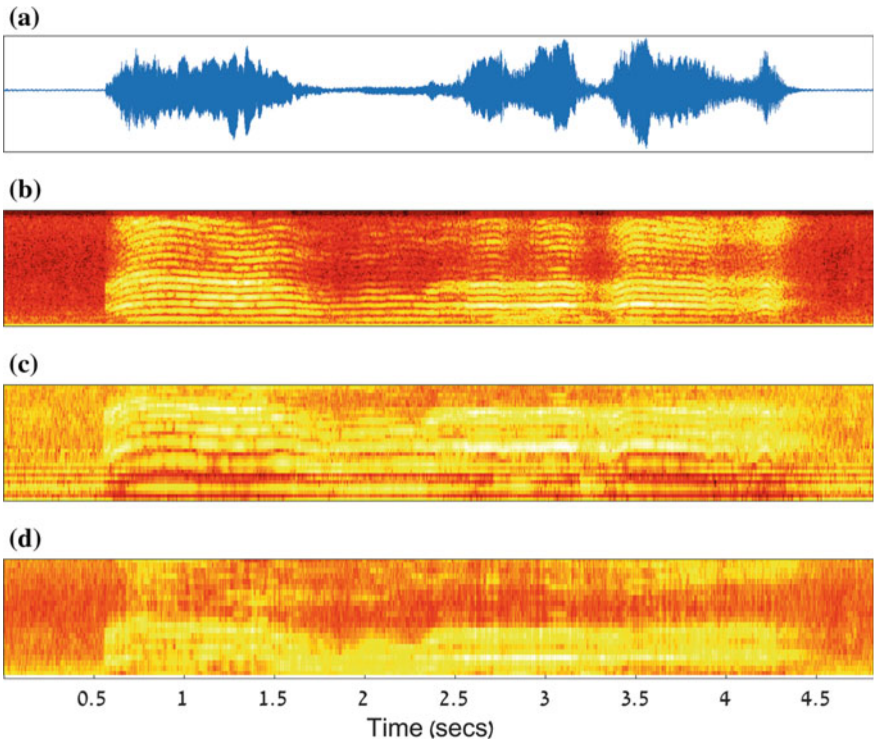


Fig. 1 Different representations of a short segment (5 s) of a baby cry signal. **a** Waveform of the signal; **b** Spectrogram; **c** Mel Filter Bank (MFB) representation; **d** Linear Filter Bank (LFB) representation. In **b**, **c**, and **d** the vertical axis represents the frequency axis in a range of 0–5 kHz. Note that in **b** (spectrogram) and **d** (LFB) the frequency axis is linear, while in **c** it is logarithmic (mel-frequency)

2.2 *Feed-Forward Architectures*

In natural images, both image dimensions carry the same content (pixel color values). Therefore, CNNs for images typically use two-dimensional filters that share weights across both dimensions and symmetric (e.g., 3 $\times$ 3) kernels. However, in the audio domain, a crucial observation is that for time-frequency representations, the x and y axes represent fundamentally different units, i.e., time in seconds and frequency in Hz. In addition, the scale of each axis might be different. Taking the LMFB representation as an example, the frequency axis is in logarithmic scale whereas the time axis is in linear scale. This calls for a careful design of the filter kernels, preferably concentrating on frequency rather than time content.

In [29], we developed a specialized CNN architecture for cry detection. Most notably, we used convolution layers with “tall” filters, i.e., non-symmetric kernels with height (frequency content) larger than width (time content). This choice of kernels is motivated by the logarithmic scale of the frequency in the LMFB representation. The use of “tall” filters makes the network “focus” on the frequency behaviour, better capturing subtle changes in signals with similar frequency content.

In this study, we improve the architecture proposed in [29] and compare the results to an architecture based on the Inception module [68]. In our CNN architecture, we have five convolutional layers followed by a fully-connected layer for final classification. We start with a 14×10 kernel for the first convolutional layer, reducing each dimension of the kernel by 2 for each subsequent layer. This gradual decrease of the kernel dimensions can be seen as multi-scale processing of the input data, which captures the frequency behaviour in an efficient manner. The architecture is presented in Table 1. Note that each convolutional layer is followed by ReLU (omitted in Table 1).

In our experiments, we were looking to test whether our specialized “tall” kernels perform better than the more common 3×3 kernels. In addition, we were interested in architectures with a smaller number of trainable parameters. For this purpose, we considered two additional architectures similar to those presented in Table 1: the first architecture has kernels replaced by 3×3 kernels for all convolutional layers; the second has the same “tall” kernels, but with a reduced number of filters, such that the total number of trainable parameters is 270,000. A visualization of the feature maps obtained for the first, third and fifth convolutional layers after the training phase is provided in Figs. 2 and 3. Note that the differences in scale of the feature maps between the figures are due to the different kernel sizes. This visualization demonstrates that the learned features in each architecture are significantly different. As expected, the features in the first layer are less structured, whereas the last features exhibit more organized patterns.

For comparison, we used an architecture based on modules similar to the Inception-ResNet-A module used in the *Inception-Resnet-v2* network [68]. In the Inception-ResNet-A module, the input is processed through two parallel convolution paths, composed of convolutional layers with different kernel sizes. For improved performance, a residual connection [69] is used, so that the output of the convolution

Table 1 Our CNN 9.6M architecture

Layer	Filter size, #filters	Activations	#Parameters
Conv1	14×10,300	$37 \times 20 \times 300$	42,300
Conv2	12×8,250	$26 \times 13 \times 250$	7,200,250
Conv3	8×6,250	$19 \times 8 \times 150$	1,800,150
Conv4	6×4,150	$14 \times 5 \times 150$	540,150
Conv5	4×2,50	$11 \times 4 \times 50$	60,050
Fully-connected	–	2	4,402
Total	–	–	9,647,302

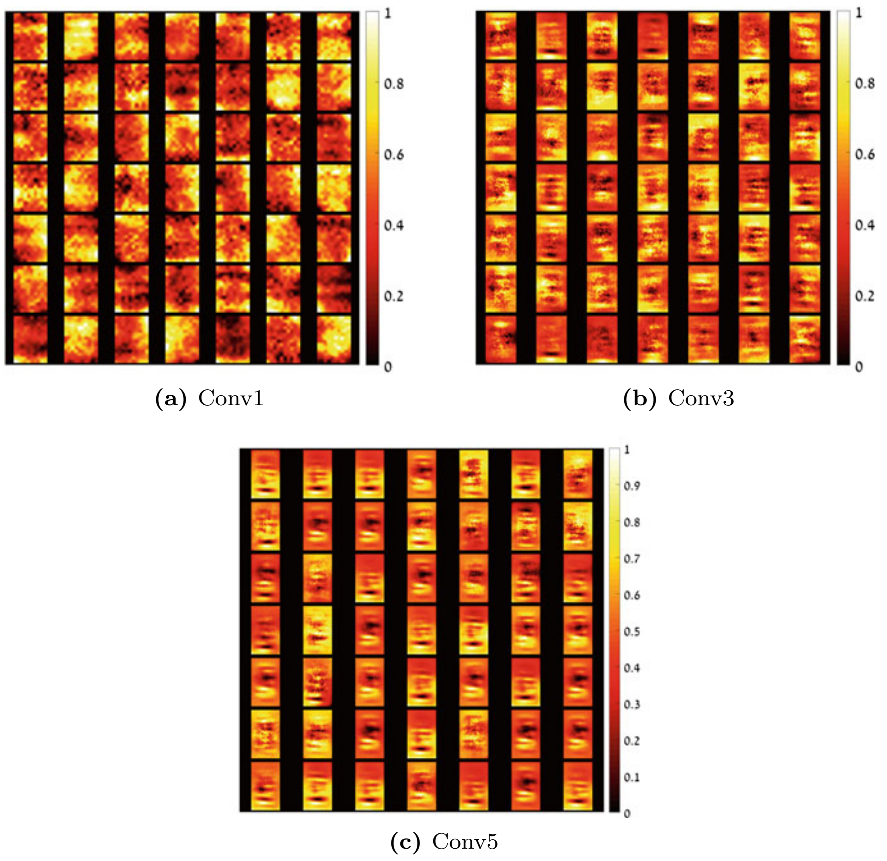
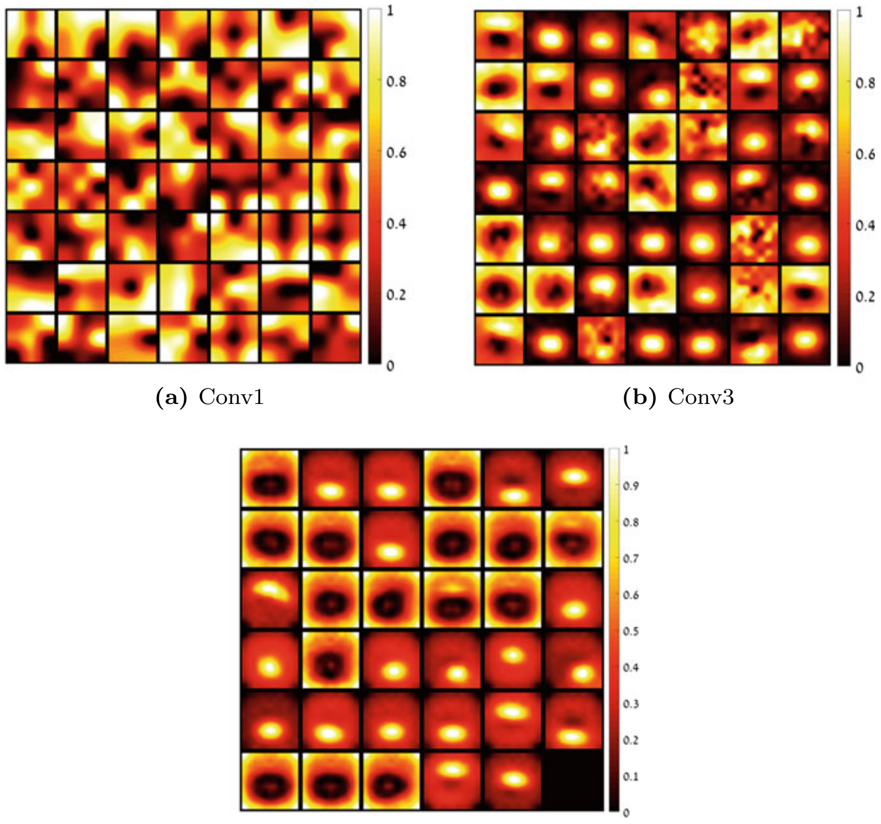


Fig. 2 A visualization of the feature maps (scaled) produced by our CNN architecture with “tall” kernels. Note that only the first 49 maps are shown for each layer

operation of the inception module is added to the input. In addition, batch normalization [70] is applied to improve training convergence. To match the input and output depth size, 1×1 kernels are applied to both the input and its processed version. The inception architecture is relatively fast to train, as it is mostly based on convolutional layers.

In our variation of Inception-ResNet-A module we omit the last 384 filters of 1×1 kernels for reduced complexity and due to the relatively low dimensionality of the input. In our experiments, we consider up to three Inception-ResNet-A modules. To obtain a proper two-class (cry/not cry) distribution, we reduce the output depth of the last module to 2 using two fully-connected layers (with 10 and 2 units, respectively). The final output is obtained by applying softmax to the output of the last fully-connected layer.



(a) Conv1 **(b)** Conv3

(c) Conv5

Fig. 3 A visualization of the feature maps produced by our CNN architecture with 3×3 kernels

2.3 Recurrent Neural Networks (RNNs)

In traditional feed-forward architectures, the inputs and the outputs are independent of each other. As a result, such networks are not capable of modeling sequences; for example, feed-forward networks might not be an optimal choice for tasks such as predicting words in a sentence (e.g., auto-complete), as we need to know what the previous words were. On the other hand, recurrent neural networks (RNNs) [65] are designed to capture temporal information, by introducing a memory component. In recent years, RNNs had great success in a variety of problems such as language modeling, translation, image captioning and more. They have also been found useful for tasks of audio detection and classification [71], in particular for speech recognition [72].

In RNNs, the output at each time instant depends on previous computations. This is obtained by learning a *state* for each time instant, which depends on the current input and the previous state. The initial state is typically initialized to all zeroes. Compared to a possible approach of using 3D convolutions (i.e., operating on the temporal axis as well), RNNs with 2D convolutions (and states) offer a more efficient and less complex approach for learning spatiotemporal features. For our application of cry detection, the use of memory is expected to be beneficial, as cry sequences are likely to be correlated.

In our experiments, we studied the performance of *bidirectional* recurrent neural network (BiRNN) architectures [73]. In BiRNNs, the output at each time instant depends on future inputs as well as on past ones: the network has both *forward* and *backward* states, which are used at each time instant to compute an output. BiRNNs typically exhibit improved performance and convergence behaviour over standard one-sided RNNs [72, 74, 75]. We considered two kinds of BiRNN architectures, each with two layers. In the first architecture, we used standard BiRNN layers. In our second architecture, we replaced the BiRNN layers with bidirectional long short-term memory (BiLSTM) layers [76–78], in which the computation of the state is more complex compared to BiRNN, resulting in better capturing of long-term dependencies. In addition, BiLSTM layers are less susceptible to the problem of vanishing or exploding gradients [79].

As a preliminary step, the input data is processed by an all-convolutional network, with the same structure (apart from the fully-connected layer) as in Table 1, but with only 270,000 parameters. This number of parameters is obtained by reducing the number of filters in each layer of the architecture presented in Table 1 by an appropriate factor. In both our BiRNN and BiLSTM architectures, we used states with 128 units.

3 Classical Approaches

Classical approaches for baby cry detection typically involve extraction of distinguishing features from segments of the audio signal, and using them to train a classifier. Common features include pitch, formants, and various temporal and spectral properties, such as short-time energy, Mel-frequency cepstrum coefficients (MFCC) and others [28, 37, 80]. In our study we compared the deep learning architectures to two traditional techniques–one using a logistic regression classifier, and the other using a Support Vector Machine (SVM). The duration of the audio segments and the size of the feature vectors, were similar to those used in the deep learning algorithms.

3.1 *Preprocessing and Feature Extraction*

The audio recordings are divided into consecutive overlapping segments of 4096 samples (about 93 ms) with an overlap of 50%. For pitch detection purposes, the segments are further divided into frames of 16 ms.

The following features are computed for each audio segment:

1. **Pitch-related features**. The pitch detection algorithm uses peaks in the cepstrum domain $c(n) = \texttt{IDFT}(\log(\texttt{DFT}|x(n)|))$ to obtain a rough estimation, and cross-correlation in the time-domain for refinement of the initial pitch value [81]. Once the prominent peak N_p is found in the section of a cepstrum that corresponds to the expected periodicity in baby cry signals (200–600 Hz), a more accurate value is obtained by finding the maximum cross-correlation between adjacent signal vectors of length K, where K is a value in a neighborhood of size δ around N_p. This approach is based on the assumption that maximal similarity between the vectors is obtained when their length is equal to the pitch period N_0:

$$N_0 = \underset{N_p-\delta \le K \le N_p+\delta}{\operatorname{argmax}} \frac{\sum_{j=1}^{K} y_1(j) \cdot y_2(j)}{\sqrt{\sum_{j=1}^{K} y_1^2(j) \cdot \sum_{j=1}^{K} y_2^2(j)}} \tag{1}$$

If the maximum cross-correlation is over 0.85, the pitch is considered valid and the frame is considered voiced. Since the pitch is computed on a frame level, the following segment level features are extracted from it: the *median value* across the segment, and *run-length* (number of consecutive voiced frames).

2. **Harmonics analysis**. Cry bursts are predominantly voiced, and therefore their signal is characterized by a harmonic structure, as demonstrated in Fig. 4. Therefore, we expect high spectral energy content around the harmonic frequencies, and compute two parameters to capture that. The first, known as *Harmonicity Factor* (H_f) measures the harmonic content of a given frame by finding the frequencies of the L most prominent peaks f_i in the spectrum, and calculating the amount of their deviation from a predicted harmonic frequency according to the corresponding f_0 (pitch) estimation, as follows:

$$H_f = \frac{1}{L} \sum_{i=1}^{L} \min(f_i \bmod f_0,\ f_0 - f_i \bmod f_0) \tag{2}$$

For harmonic peaks the distance between the frequency and an integer multiple of f_0 should be small, and therefore for frames with harmonic structure H_f should be small (see Fig. 5).
The second parameter is the *Harmonic-to-Average Power Ratio* (HAPR) which measures the ratio between the power of the first M harmonic components to that of the average spectral power of the frame. If $X(k)$ is the DFT of the audio frame, and f_m is the frequency of peak in the vicinity of the mth harmonic, *HAPR* is computed as follows [82, 83]:

$$\text{HAPR} = \frac{1}{M}\sum_{m=2}^{M} 10\log_{10}\frac{|X(f_m)|^2}{(\frac{1}{N}\sum\limits_{k=0}^{N-1}|X(2\pi k/N)|^2)} \tag{3}$$

3. **Filter banks and cepstrum coefficients**. A filter-bank representation is obtained by multiplying the power spectrum $X_m(K)$ by triangularly-shaped filters $V_i(K)$, where U_i and L_i are the lower and upper bounds of each filter, respectively, and $S_i = \sum_{K=U_i}^{L_i} |V_i(K)|^2$ is a normalization coefficient to compensate for the variable bandwidth of the filters:

$$E_i = \frac{1}{S_i}\sum_{K=U_i}^{L_i} |X_m(K)V_i(K)|^2 \tag{4}$$

   A commonly used filter-bank representation uses the Mel scale, which arranges the filters and their widths in a way that attempts to mimic the auditory perception of the human ear [84, 85]. A comparison between the Mel filter-bank (MFB) and Linear filter-bank (LFB) is shown in Fig. 6, and their spectral representations can also be seen in Fig. 1.
   When MFB representation is used, the Mel-Filter Cepstrum Coefficients (MFCC) vector is obtained by applying the Discrete Cosine Transform (DCT) to the logarithm of the energy vector E_i. In our algorithm, we used the first 38 MFCC.
   Figure 7 shows an example of the distribution of the 5th MFC coefficient among baby cry sections (red) versus all other sound events (blue) in the training set (about 320 s). The discriminating potential of this feature is evident, although there is a wide overlapping area.
4. **Spectral energy parameters**. Different types of audio signals often exhibit different patterns in the spectral energy, as represented by the power spectrum $|X_m(K)|^2$. We computed the following two parameters: the *Spectrum rolloff point* f_R—the frequency below which 75% of the spectral energy is concentrated, and the *Band energy ratio* between the total spectral energies of two frequency bands [0, 2500 Hz] and [2500 Hz, $F_s/2$], where F_s is the sampling frequency:

$$f_R = f : \sum_{K=0}^{f} |X_m(K)|^2 = 0.75 \cdot \sum_{K=0}^{F_s/2} |X_m(K)|^2 \tag{5}$$

$$\text{BER} = 10\log_{10}\frac{\sum_{K(f_{2500\,\text{Hz}})}^{F_s/2} |X_m(K)|^2}{\sum_{K=0}^{f_{2500\,\text{Hz}}} |X_m(K)|^2} \tag{6}$$

5. **Time-domain features**. Simple time-domain features, such as the zero-crossing rate (ZCR) and short-time energy were also used:

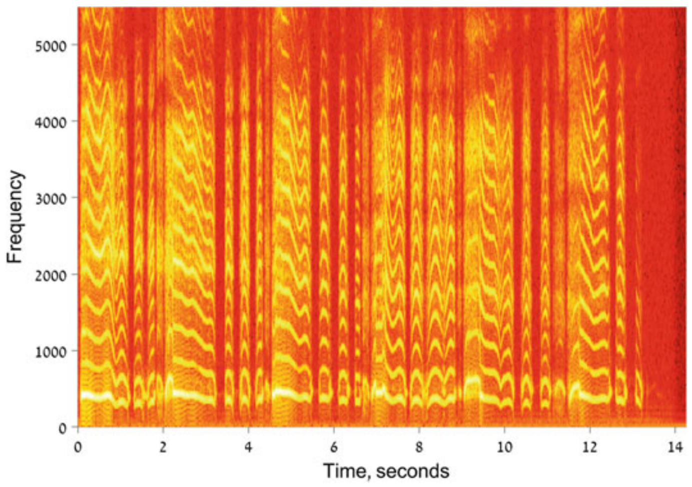


Fig. 4 A spectrogram of an infant cry signal, demonstrating the harmonic structure of voiced bursts

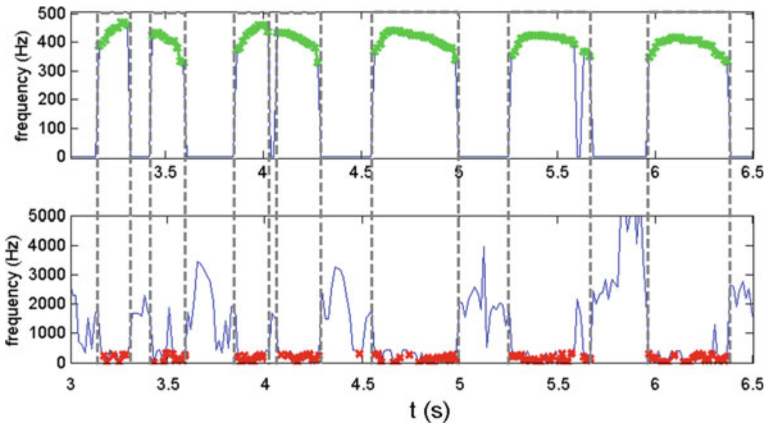


Fig. 5 Fundamental frequency F_0, and Harmonicity Factor (H_f) aligned for several cry bursts

$$\mathrm{ZCR}(m) = \frac{1}{2N}\sum_{n=1}^{N-1} |\mathrm{sign}(x_m(n)) - \mathrm{sign}(x_m(n-1))|. \tag{7}$$

$$E(m) = \frac{1}{N}\sum_{n=0}^{N-1} x_m^2(n). \tag{8}$$

A detailed description of the features and their computation is available in [9, 67, 80].

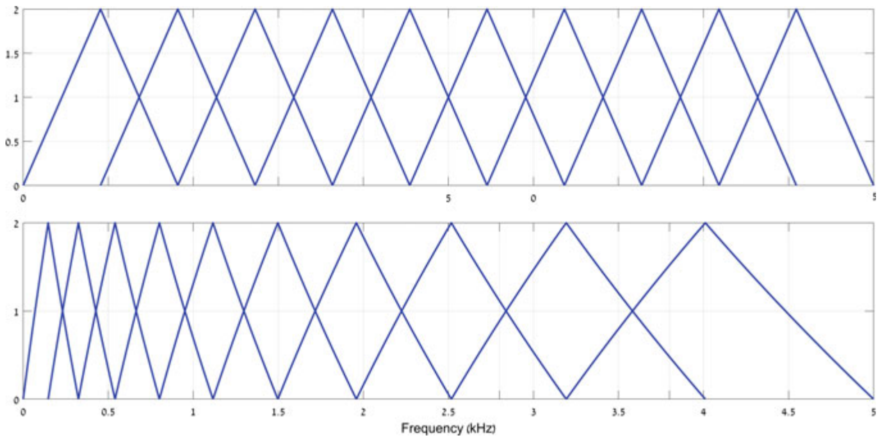


Fig. 6 A schematic description of Mel-filter bank (bottom) versus Linear filter bank (top), both with 10 filters in the range of 0–5 kHz

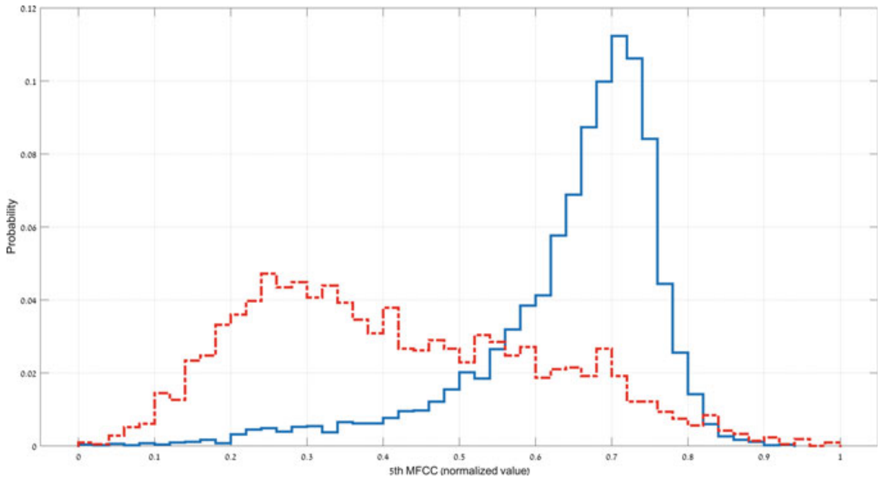


Fig. 7 A histogram of the 5th MFC coefficient. Dashed line: cry events, solid line: other events

3.2 Logistic Regression

The *logistic regression* classifier [64] is a simple supervised discriminative algorithm, with low computational complexity. The logistic regression is a non-linear hypothesis function of the form:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-\boldsymbol{\theta}^T \boldsymbol{x}\right)}, \tag{9}$$

where $\boldsymbol{x}$ is a d-dimensional feature vector and $\boldsymbol{\theta}$ is a weight vector. In our case, $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \in (0, 1)$ predicts the likelihood of a segment to be a cry sound (values close to 1), or a different sound (values close to 0). The final binary classification $y \in \{0, 1\}$ (where 1 denotes a cry event) is obtained by comparing $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \in (0, 1)$ to a threshold value. In the training phase of the classifier, a gradient descent algorithm is used to find $\boldsymbol{\theta}$ that minimizes the (L_2) regularized *cross-entropy cost function*

$$\begin{aligned}\mathbb{E}(\boldsymbol{\theta}) = &-\frac{1}{n}\sum_{j=1}^{n} y^{(j)} \log\left(\frac{1}{1+\exp(-\boldsymbol{\theta}^T\boldsymbol{x}^{(j)})}\right)\\ &-\frac{1}{n}\sum_{j=1}^{n}\left(1-y^{(j)}\right)\log\left(\frac{\exp(-\boldsymbol{\theta}^T\boldsymbol{x}^{(j)})}{1+\exp(-\boldsymbol{\theta}^T\boldsymbol{x}^{(j)})}\right)\\ &+\frac{\lambda}{2n}\sum_{k=1}^{d}{\theta_k}^2,\end{aligned} \tag{10}$$

given a dataset of n labeled samples $\left\{\boldsymbol{x}^{(j)}, y^{(j)}\right\}_{j=1}^{n}$, where λ is a regularization parameter. The $\boldsymbol{\theta}$-minimizer found by the stochastic gradient descent algorithm is then assigned to (9) to classify new unlabeled samples.

A schematic block diagram of the logistic-regression-based algorithm is shown in Fig. 8. The input data is divided into consecutive segments of 4096 samples. For each segment a 50-dimensional feature vector is computed. The trained regularized logistic regression is then applied to each feature vector, and the hypothesis function $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ is obtained, representing an estimation of the posterior probability $p(y|\boldsymbol{x})$, where $y \in \{0, 1\}$ is the sound event to be classified as cry or non-cry and $\boldsymbol{x}$ is the feature vector. Using a threshold value $\mathtt{Th}_1$, an initial decision value for each segment is set according to the following rule:

$$d(n) = \begin{cases} 1, & \text{if } h_{\boldsymbol{\theta}}(\boldsymbol{x}) > \mathtt{Th}_1 \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

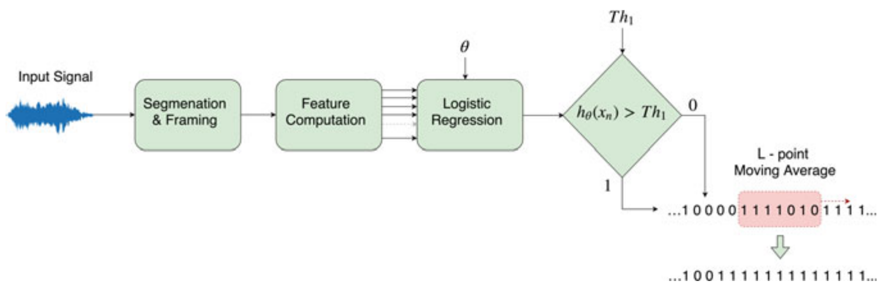


Fig. 8 A schematic block diagram of the logistic regression algorithm

The duration of a single segment is about 93 ms, while most cry events are at least several hundred of milliseconds long. In order to avoid erroneous detection of sections that are too short to be a likely cry event, a smoothing operation is performed as follows: a sliding window of length L is applied to the initial sequence of decisions and the smoothed decision $d_s(n)$ for the central segment is updated according to the following rule:

$$d_s(n) = \begin{cases} 1, & \text{if } \sum_{k=-M}^{M} d(n-k) > \mathtt{Th}_2 \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

where L is odd, $M = (L-1)/2$ and $\mathtt{Th}_2 \in [1, L]$ is a predefined threshold value.

3.3 Support Vector Machine

The Support Vector Machine (SVM) is a supervised large-margin classifier, which atteempts to find a separating hyperplane between two classes of data, such that the margin between the hyperplane and the closest samples in either set is maximized. To train the SVM, we applied the Sequential Minimal Optimization (SMO) algorithm [86], which solves the following minimization problem:

$$\min_{\theta_0,\theta,\xi} \frac{1}{2} \|\theta\| + C \sum_{m=1}^{M} \xi_m \;\; s.t. \;\; x_m \geq 0, y_m(\theta^T x_m + \theta_0) \geq 1 - \xi_m, \quad m = 1, 2, ..., M \tag{13}$$

where θ and θ_0 are parameters of the maximal margin hyperplane to be learned, x_m are the input data points (feature vectors), $y_m \in \{-1, 1\}$ are the output points (data labels - “cry” or “not cry”), ξ are slack variables that permit margin failure and C trades off a small number of margin failures and wide margin [86].

The trained SVM can be used in place of the logistic regression classifier in the block diagram of Fig. 8. Performance comparison of the two classifiers is given in Table 2.

Table 2 A summary of the false-positive rates for a given detection rate

Classifier/Detection rate	75%	80%	85%	90%	95%
CNN 9.6M “tall”	0.25%	0.44%	0.66%	1.07%	3.50%
CNN 270K “tall”	0.47%	0.72%	0.95%	1.95%	6.30%
CNN 9.6M 3 × 3	0.51%	0.73%	1.30%	2.39%	5.41%
CNN 270K 3 × 3	0.48%	0.98%	2.01%	3.22%	7.8%
SVM	0.31%	0.52%	1.39%	4.13%	9.75%
Logistic regression	0.38%	0.6%	1.23%	3.81%	9.80%

4 Performance Evaluation

4.1 Database

The database for this study consists of three hours of audio recordings (sampled at 44, 100 Hz) of 0–6 month old babies in the Netherlands using off-the-shelf smartphones [87]. The babies were recorded continuously for several days in a domestic environment. The recordings were fully annotated, with about 50 different event types, such as crying, parents talking, door opening/closing, etc.

4.2 Training and Test Process

Our training corpus contained 14% of the labeled data, whereas the test corpus contained the remaining 86%. We trained our feed-forward CNN architectures using MATLAB and our RNN architectures (BiRNN and BiLSTM) with TensorFlow. We used Adam [88], which is an adaptive learning rate optimization algorithm, with an initial learning step of 0.00001. The gradient in each iteration was evaluated using mini-batches of 32 segments. Our loss function was cross-entropy loss. To avoid over-fitting, we applied L2 regularization to the weights, with a scale of 0.0001. The networks were trained over 20 epochs of the training data. The hardware used includes Intel Core i7-7700K 4.2 GHz CPU and NVIDIA GTX 1080Ti 11GB GDDR5 GPU.

During testing (i.e., inference), we used the majority vote process depicted in Sect. 3.2 with $L = 17$. That is, a segment was classified as ‘cry’ if at least 8 other segments in a neighbourhood of 17 segments were classified as ‘cry’. The measured inference latency (in software) for CNN 9.6M was smaller than 2.5 ms per segment. In fact, this latency can be considerably reduced by using dedicated tools for deploying trained networks in hardware, such as TensorRT by NVIDIA or Deep Learning Deployment Kit by Intel.

4.3 Results

As performance metrics, we used two important measures known as the *detection rate* and the *false-positive rate*. The detection rate (also known as *sensitivity* or *recall*) is defined as the ratio between the number of true-positive events (TP), i.e. the number of cry events correctly identified, and the total number of cry events in the recording set (true positives TP and false negatives FN). The false-positive (or *false-alarm*) rate is defined as the ratio between the number of false positives (non-cry events identified erroneously as cry events (FP)) and the total number of non-cry events in the recording set (false positives (FP) and true negatives (TN)). The detection rate is therefore $\text{TP}/(\text{TP} + \text{FN})$, and the false-positive rate is $\text{FP}/(\text{FP} + \text{TN})$.

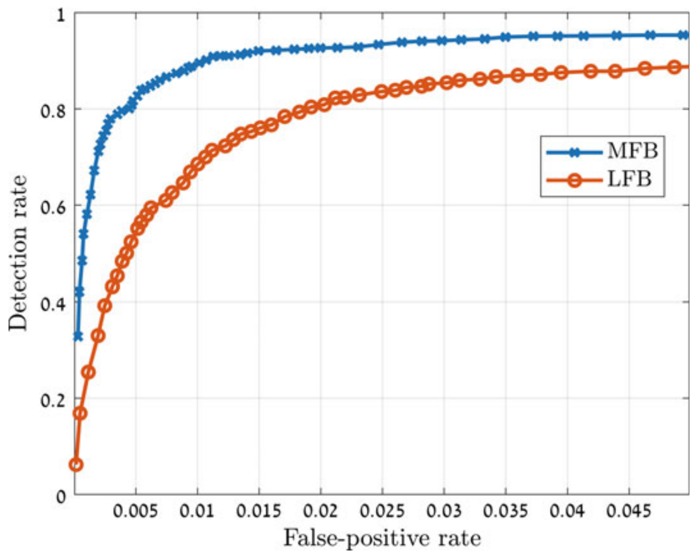


Fig. 9 A comparison between MFB and LFB ROC curves for the CNN with 9.6M parameters

In the analysis of the cry-detection performance of different classifiers we focus on the trade-off between the false-positive rate and the detection rate. In particular, we are interested in the performance in the low false-positive rate regime, which is required in practice. The performance evaluation was carried out using receiver operating characteristic (ROC) curves.

First, we were interested in comparing the MFB and LFB representations (see Sect. 2.1), by comparing the ROC curves of our CNN 9.6M architecture for both representations. As shown in Fig. 9, the MFB representation is superior to LFB. As discussed earlier, this is expected as the Mel-scale in MFB is better suited for detecting signals in the presence of noise with similar characteristics.

In our next experiment, we compared the performance of the feed-forward architectures described in Sect. 2.2. The results are presented in Fig. 10. First, for all false-positive rates, our CNN 9.6M "tall" architecture outperforms all other architectures. This is most noticeable for false-positive rates below 2%. It is interesting to note that the CNN 270K "tall" network has very good performance, despite using far fewer parameters compared to the CNN 9.6M "tall" architecture. In addition, it has similar results to CNN 9.6M 3×3. That is, the use of "tall" filters is shown to be highly beneficial, even for networks with a small number of parameters.

We later compared our CNN 9.6M to the BiRNN and BiLSTM architectures described in Sect. 2.3. The results are shown in Fig. 11. As evident by the result, the introduction of memory through the use of RNN has limited impact. The reason might be the short duration of cry events, which limits the benefit of architectures with memory. This perhaps hints as well that a more powerful CNN for feature extraction in addition to a longer training process are required for such architectures.

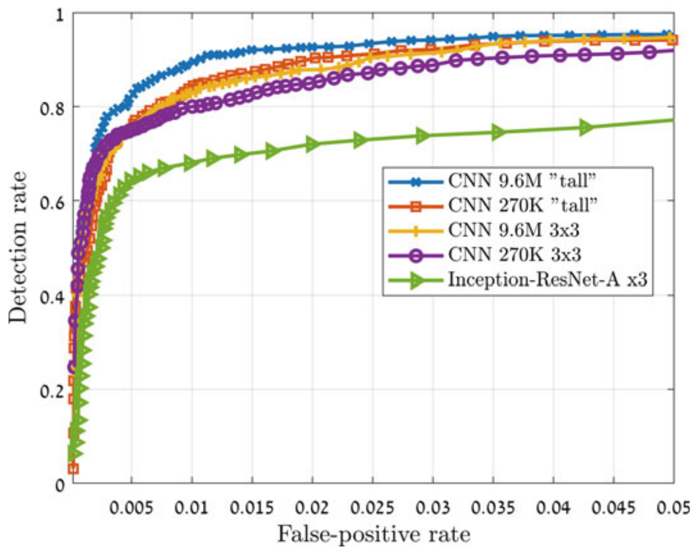


Fig. 10 ROC curves of our feed-forward CNN architectures

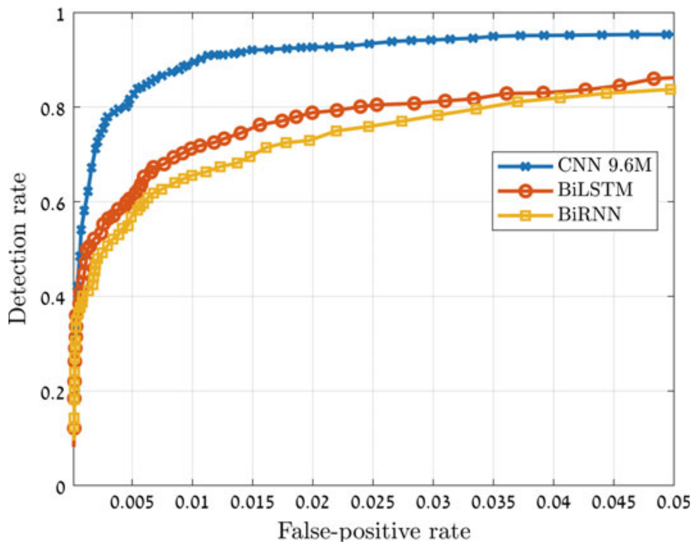


Fig. 11 A comparison between CNN with 9.6M parameters, BiLSTM and BiRNN

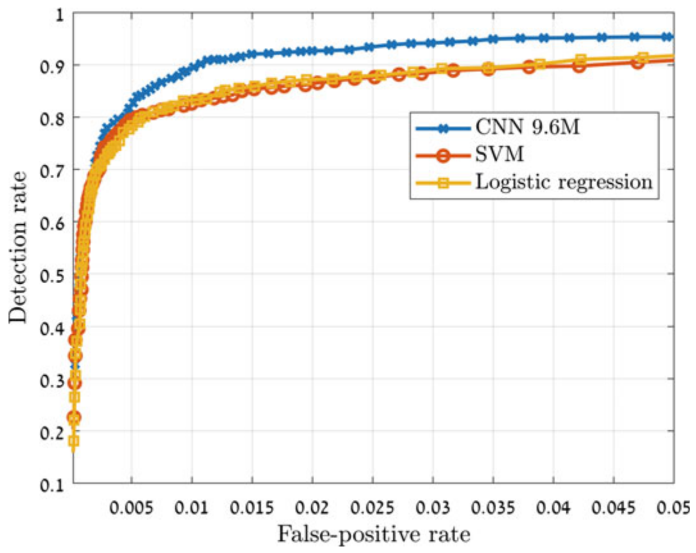


Fig. 12 ROC curves for the CNN, SVM and logistic regression classifiers

Finally, we compared our CNN 9.6M "tall" architecture to logistic regression and SVM. This is shown in Fig. 12. We see that the deep learning approach outperforms the traditional classifiers, though the performance of the latter is quite good, and at very low false-positive regimes it even outperforms CNN architectures "weaker" than CNN 9.6M "tall".

The evaluation results are summarized in Table 2. For detection rates of 75, 80, 85, 90 and 95%, the false-positive rates of the CNN 9.6M "tall" classifier are the lowest, compared to other deep-learning approaches as well to conventional machine learning algorithms such as the SVM or logistic regression.

5 Conclusion

In this study, we evaluated the performance of both deep learning and traditional approaches for baby cry detection. We investigated several CNN architectures, as well as recurrent neural networks (including LSTM) for better capturing temporal behaviour. We studied image representations of the input audio signals and found the log Mel-filter Bank (MFB) to be an appropriate representation. We demonstrated that by carefully choosing the kernel sizes and shapes in accordance to the MFB representation, better performance is achieved compared to common deep learning architectures. Our CNN classifier was shown to yield considerably better results compared to a traditional machine learning classifiers such as SVM or logistic regression, especially for low false-positive rates (which is highly required in practice). Our study demonstrates the power and advantages of deep learning when applied to audio event detection.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105. Curran Associates, Inc. (2012). http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
2. Hershey, S., Chaudhuri, S., Ellis, D.P., Gemmeke, J.F., Jansen, A., Moore, R.C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B.: CNN architectures for large-scale audio classification. In: IEEE International Conference on Acoustics, Speech and Signal Processing (icassp), pp. 131–135. IEEE (2017)
3. Cakir, E., Heittola, T., Huttunen, H., Virtanen, T.: Polyphonic sound event detection using multi label deep neural networks. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE (2015)
4. Ramírez, J., Górriz, J.M., Segura, J.C.: Voice Activity Detection. Fundamentals And Speech Recognition System Robustness (2007)
5. Ruinskiy, D., Lavner, Y.: An effective algorithm for automatic detection and exact demarcation of breath sounds in speech and song signals. IEEE Trans. Audio Speech Lang. Process. **15**, 838–850 (2007)
6. Kong, Y.-Y., Mullangi, A., Kokkinakis, K.: Classification of fricative consonants for speech enhancement in hearing devices. PloS one (2014)
7. Frid, A., Lavner, Y.: Spectral and textural features for automatic classification of fricatives. In: XXII Annual Pacific Voice Conference (PVC), pp. 1–4 (2014)
8. Panagiotakis, C., Tziritas, G.: A speech/music discriminator based on rms and zero-crossings. IEEE Trans. Multimed. **7**, 155–166 (2005)
9. Lavner, Y., Ruinskiy, D.: A decision-tree-based algorithm for speech/music classification and segmentation. EURASIP J. Audio Speech Music Process. (2009)
10. Tzanetakis, G., Cook, P.: Musical genre classification of audio signals. IEEE Trans. Speech Audio Process. **10**(5), 293–302 (2002)
11. Barchiesi, D., Giannoulis, D., Stowell, D., Plumbley, M.D.: Acoustic scene classification: classifying environments from the sounds they produce. IEEE Signal Process. Mag. **32**(3), 16–34 (2015)
12. Morgan, N., Bourlard, H.: Continuous speech recognition. IEEE Signal Process. Mag. **12**(3), 24–42 (1995)
13. Aruna, C., Parameswari, A.D., Malini, M., Gopu, G.: Voice recognition and touch screen control based wheel chair for paraplegic persons. In: 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), pp. 1–5 (2014)
14. Carletti, V., Foggia, P., Percannella, G., Saggese, A., Strisciuglio, N., Vento, M.: Audio surveillance using a bag of aural words classifier. In: 2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance, pp. 81–86 (2013)
15. Ye, J., Kobayashi, T., Higuchi, T.: Audio-based indoor health monitoring system using flac features. In: 2010 International Conference on Emerging Security Technologies, pp. 90–95 (2010)
16. Kawano, D., Ogawa, T., Matsumoto, H.: A proposal of the method to suppress a click noise only from an observed audio signal. In: 2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp. 93–96 (2017)

17. Zhang, H., McLoughlin, I., Song, Y.: Robust sound event recognition using convolutional neural networks. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 559–563 (2015)
18. Piczak, K.J.: Environmental sound classification with convolutional neural networks. In: IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6 (2015)
19. Valenti, M., Diment, A., Parascandolo, G., Squartini, S., Virtanen, T.: DCASE 2016 acoustic scene classification using convolutional neural networks. In: Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016). Tampere University of Technology. Department of Signal Processing (2016)
20. Çakır, E., Parascandolo, G., Heittola, T., Huttunen, H., Virtanen, T.: Convolutional recurrent neural networks for polyphonic sound event detection. IEEE/ACM Trans. Audio Speech Lang. Process. **25**(6), 1291–1303 (2017)
21. Naithani, G., Barker, T., Parascandolo, G., Bramslw, L., Pontoppidan, N.H., Virtanen, T.: Low latency sound source separation using convolutional recurrent neural networks. In: 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 71–75 (2017)
22. Dieleman, S., Schrauwen, B.: End-to-end learning for music audio. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6964–6968 (2014)
23. Pons, J., Nieto, O., Prockup, M., Schmidt, E.M., Ehmann, A.F., Serra, X.: End-to-end learning for music audio tagging at scale. In: ISMIR (2018)
24. Ferretti, D., Severini, M., Principi, E., Cenci, A., Squartini,S.: Infant cry detection in adverse acoustic environments by using deep neural networks. In: EUSIPCO (2018)
25. Turan, M.A.T., Erzin, E.: Monitoring infant's emotional cry in domestic environments using the capsule network architecture. In: Interspeech (2018)
26. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: NIPS (2017)
27. Torres, R., Battaglino, D., Lepauloux, L.: Baby cry sound detection: a comparison of hand crafted features and deep learning approach. In: EANN (2017)
28. Saraswathy, J., Hariharan, M., Yaacob, S., Khairunizam, W.: Automatic classification of infant cry: a review. In: 2012 International Conference on Biomedical Engineering (ICoBE), pp. 543–548 (2012)
29. Lavner, Y., Cohen, R., Ruinskiy, D., IJzerman, H.:Baby cry detection in domestic environment using deep learning. In: 2016 International Conference on the Sceience of Electrical Engineering (ICSEE 2016) (2016)
30. Zhang, X., Zou, Y., Liu, Y.: AICDS: An Infant Crying Detection System Based on Lightweight Convolutional Neural Network, pp. 185–196. Springer (2018)
31. Xu, Y., Hasegawa-Johnson, M., McElwain, N.: Infant emotional outbursts detection in infant-parent spoken interactions. In: Interspeech (2018)
32. Silva, G., Wickramasinghe, D.: Infant cry detection system with automatic soothing and video monitoring functions. J. Eng. Technol. Open Univ. Sri Lanka (JET-OUSL) **5**(1). http://digital.lib.ou.ac.lk/docs/handle/701300122/1476 (2017)
33. Gao, J., Pabon, L.: Hot car baby detector. Illinois College of Engineering, Technical Report, December 2014
34. Lollipop smart baby monitor. https://www.lollipop.camera/ (2018)
35. Cocoon cam baby monitor. https://cocooncam.com/ (2019)
36. Evoz wifi baby vision monitor. https://myevoz.com/ (2019)
37. Varallyay, G.: The melody of crying. Int. J. Pediatr. Otorhinolaryngol. **71**(11), 1699–1708 (2007)
38. Zabidi, A., Khuan, L.Y., Mansor, W., Yassin, I.M., Sahak, R.: Classification of infant cries with asphyxia using multilayer perceptron neural network. In: Proceedings of the 2010 Second International Conference on Computer Engineering and Applications—Series. ICCEA 2010, vol. 01, pp. 204–208. IEEE Computer Society, Washington, DC, USA (2010)
39. Orlandi, S., Reyes-Garcia, C.A., Bandini, A., Donzelli, G., Manfredi, C.: Application of pattern recognition techniques to the classification of full-term and preterm infant cry. J. Voice Off. J. Voice Found. **30**, 10 (2015)

40. Michelsson, K., Michelsson, O.: Phonation in the newborn, infant cry. Int. J. Pediatr. Otorhinolaryngol. **49**, S297–S301 (1999)
41. Bowlby, J.: Attachment and Loss. Basic Books, vol. 1 (1969)
42. Ostwald, P.: The sounds of infancy. Dev. Med. Child Neurol. **14**(3), 350–361 (1972)
43. Owings, D., Zeifman, D.: Human infant crying as an animal communication system: insights from an assessment/management approach. In: Evolution of Communication Systems: A Comparative Approach, pp. 151–170 (2004)
44. Nelson, J.: Seeing Through Tears: Crying and Attachment. Routledge (2005)
45. IJzerman, H., et al.: A theory of social thermoregulation in human primates. Front. Psychol. **6**, 464 (2015)
46. Butler, E.A., Randall, A.K.: Emotional coregulation in close relationships. Emot. Rev. **5**(2), 202–210 (2013)
47. LaGasse, L.L., Neal, A.R., Lester, B.M.: Assessment of infant cry: a coustic cry analysis and parental perception. Ment. Retard. Dev. Disabil. Res. Rev. **11**(1), 83–93 (2005)
48. Hendriks, M., Nelson, J.K., Cornelius, R., Vingerhoets, A.: Why crying improves our well-being: an attachment-theory perspective on the functions of adult crying. In: Emotion Regulation: Conceptual and Clinical Issues, pp. 87–96 (2008)
49. Pal, P.A., Iyer, N., Yantorno, R.E.: Emotion detection from infant facial expressions and cries. In: 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 2, pp. II–II (2006)
50. Barajas-Montiel, S., Reyes-Garcia, C.A.: Identifying pain and hunger in infant cry with classifiers ensembles, pp. 770 – 775 (2005)
51. Wasz-Höckert, O.: The infant cry: a spectrographic and auditory analysis. Spastics International Medical Publications in association with W. Heinemann Medical Books. Series Clinics in Developmental Medicine (1968)
52. Vingerhoets, A.: Why Only Humans Weep: Unravelling the Mysteries of Tears. Oxford University Press (2013)
53. Bell, S.M., Salter Ainsworth, M.D.: Infant crying and maternal responsiveness. In: Child development, vol. 43, pp. 1171–90 (1973)
54. Lounsbury, M.L., Bates, J.E.: The cries of infants of differing levels of perceived temperamental difficultness: acoustic properties and effects on listeners. Child Dev. **53**(3), 677–686 (1982)
55. Zeskind, P., Barr, R.: Acoustic characteristics of naturally occurring cries of infants with colic. Child Dev. **68**, 394–403 (1997)
56. Laan, A., Assen, M.V., Vingerhoets, A.: Individual differences in adult crying: the role of attachment styles. Soc. Behav. Person. Int. J. (2012)
57. Bryant Furlow, F.: Human neonatal cry quality as an honest signal of fitness. Evol. Hum. Behav. **18**, 175–193 (1997)
58. Kheddache, Y., Tadj, C.: Acoustic measures of the cry characteristics of healthy newborns and newborns with pathologies. J. Biomed. Sci. Eng. **06**(08), 796–804 (2013)
59. Orlandi, S., Manfredi, C., Bocchi, L., Scattoni, M.L.: Automatic newborn cry analysis: a non-invasive tool to help autism early diagnosis. In: 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 2953–2956 (2012)
60. Sheinkopf, S.J., Iverson, J.M., Rinaldi, M.L., Lester, B.M.: Atypical cry acoustics in 6-month-old infants at risk for autism spectrum disorder. Autism Res. **5**(5), 331–339 (2012)
61. Jeyaraman, S., Muthusamy, H., Wan, K., Jeyaraman, S., Nadarajaw, T., Yaacob, S., Nisha, S.: A review: survey on automatic infant cry analysis and classification. Health Technol. **8** (2018)
62. IJzerman, H., Čolić, M., Hennecke, M., Hong, Y., Hu, C.-P., Joy-Gaba, J., Lazarevic, D., Lazarevic, L., Parzuchowski, M., Ratner, K.G., Schubert, T., Schuetz, A., Stojilovi, D., Weissgerber, S., Zickfeld, J., Lindenberg, S.: Does distance from the equator predict self-control? Lessons from the human penguin project. Behav. Brain Sci. **40** (2017)
63. IJzerman, H., Lindenberg, S., Dalgar, I., Weissgerber, S., Clemente Vergara, R., Cairo, A., oli, M., Dursun, P., Frankowska, N., Hadi, R., Hall, C., Hong, Y., Hu, C.-P., Joy-Gaba, J., Lazarevic, D., Lazarevic, L., Parzuchowski, M., Ratner, K.G., Rothman, D., Zickfeld J.: The human penguin project: climate, social integration, and core body temperature. Collabra: Psychol. **4** (2018)

64. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer Science+Business Media (2006)
65. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, http://www.deeplearningbook.org (2016)
66. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back propagating errors. Nature **323**, 533–536 (1986)
67. Huang, X., Acero, A., Hon, H.-W.: Spoken Language Processing: A Guide to Theory, Algorithm, and System Development. Prentice Hall PTR (2001)
68. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR (2016) arXiv:1602.07261
69. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
70. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, arXiv:1502.03167
71. Phan, H., Koch, P., Katzberg, F., Maaß, M., Mazur, R., Mertins, A.: Audio scene classification with deep recurrent neural networks. In: INTERSPEECH (2017)
72. Graves, A., Mohamed, A., Hinton, G.E.: Speech recognition with deep recurrent neural networks. CoRR (2013) arXiv:1303.5778
73. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
74. Graves, A., Jaitly, N., Mohamed, A.: Hybrid speech recognition with deep bidirectional LSTM. In 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 273–278 (2013)
75. Ben-Yehuda, T., Abramovich, I., Cohen, R.: Low-complexity video classification using recurrent neural networks. In: 2018 International Conference on the Science of Electrical Engineering (ICSEE 2018) (2018)
76. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
77. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks, Series Studies in Computational Intelligence, vol. 385. Springer (2012)
78. Fei, H., Tan, F.: Bidirectional grid long short-term memory (bigridlstm): a method to address context-sensitivity and vanishing gradient. Algorithms **11** (2018)
79. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Series Proceedings of Machine Learning Research, vol. 9, pp. 249–256 PMLR (2010)
80. Cohen, R., Lavner, Y.: Infant cry analysis and detection. In: 2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel (IEEEI 2012), pp. 2–6 (2012)
81. Noll, A.M.: Cepstrum pitch determination. J. Acoust. Soc. Am. **41**(2), 293–309 (1967)
82. van Waterschoot, T., Moonen, M.: Fifty years of acoustic feedback control: state of the art and future challenges. Proc. IEEE **99**(2), 288–327 (2011)
83. van Waterschoot, T., Moonen, M.: Comparative evaluation of howling detection criteria in notch-filter-based howling suppression. J. Audio Eng. Soc. **58**(11), 923–940 (2010)
84. Rabiner, L.R., Schafer, R.W.: Theory and Applications of Digital Speech Processing, vol. 64. Pearson, Upper Saddle River (2011)
85. Quatieri, T.: Discrete-Time Speech Signal Processing: Principles and Practice. Prentice Hall, London (2002)
86. Platt, J.: Sequential minimal optimization: a fast algorithm for training support vector machines (1998)
87. Frederiks, K., Sterkenburg, P., Lavner, Y., Cohen, R., Ruinskiy, D., Verbeke, W., IJzerman, H.: Mobile social physiology as the future of relationship research and therapy: presentation of the bio-app for bonding (BAB), *PsyArXiv* (2018)
88. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations, December 2014

Securing Industrial Control Systems from False Data Injection Attacks with Convolutional Neural Networks

Sasanka Potluri, Shamim Ahmed and Christian Diedrich

Abstract Due to trends in modern infrastructure development and usage, the attacks on Industrial Control Systems (ICS) are inevitable. New threats and other forms of attacks are constantly emerging to exploit vulnerabilities in system compromising the security parameters such as Confidentiality, Integrity and Availability (CIA). Injection attacks also termed as False Data Injection Attacks (FDIA) are the complex attacks on the ICS. FDIA affects the data integrity of a packet by modifying their payloads and are considered as an intrusion via remote access. In FDIA, attackers gain access to a critical process or process parameters in ICS and forces them to execute according to the newly injected code or command. For our research, a process control plant from Integrated Automation laboratory was used to acquire different parameters related to ICS. Injection attacks such as measurement injection and command injection were simulated and injected into the obtained plant data. Convolutional Neural Networks (CNN) is used to evaluate the functionality of identifying those injection attacks. Multiple steps such as pre-processing, feature extraction, data transformation and image representation were performed in order to feed the CNN with the simulated plant data. A 3-layered fully connected CNN architecture with non-linear ReLU activation is built along with a SoftMax classification layer for classifying the input data as a normal or an attack. A proper training of CNN is done by checking the variance to avoid overfitting and underfitting of the network. Performance parameters such as accuracy, recall, precision F-measure and Cohen's kappa coefficient were computed. CNN outperforms in the performance compared to other deep learning approaches.

S. Potluri (✉) · S. Ahmed · C. Diedrich
Faculty of Electrical Engineering and Information Technology, Institute for Automation Engineering, Otto-von-Guericke University, Magdeburg, Germany
e-mail: sasanka.potluri@ovgu.de

S. Ahmed
e-mail: shamim.ahmed@ovgu.de

C. Diedrich
e-mail: christian.diedrich@ovgu.de

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_8

Keywords Convolutional neural networks · Injection attacks · Industrial control systems

1 Introduction

Industrial Control Systems (ICS) operate the industrial infrastructures world-wide including electric power, water, oil/gas, pipelines, chemicals, mining, pharmaceuticals transportation and manufacturing. ICS measure, control and provide a view of the process. Typical types of ICS include Supervisory Control and Data Acquisition (SCADA), Distributed Control Systems (DCS), Programmable Logic Controllers (PLC), Remoter Terminal Units (RTU) and field instrumentation. These types of systems are commonly utilized throughout the global industrial infrastructures. The commonality of ICS and their architecture enable the International Society of Automation (ISA) to establish one general process industry committee for cyber security—S99 [1].

ICS cyber security was formally identified in the mid-late 1990s with the publication of Presidential Decision Directive (PDD) 63 [2]. It was at that time the US Department of Energy's (DOE) National Laboratories starting performing cyber security assessments for utilities on a confidential (not classified) basis. As these assessments were not made public, there was a little knowledge of the results unless the utilities were willing to share their results. Despite several measures, several attacks took place on ICS. Some of the know attacks on ICS include but not limited to:

- Trojan attack: In 1982 it was the first know attacks on the critical infrastructure occurred in Serbia. Trojan was used to insert malicious values in pump speeds and valve settings which created high pressure beyond acceptable to the pipeline joints and welds that resulted in explosion. This attack can be considered as an injection attacks on ICS [3].
- SQLSlammer: In 2003, the SQLSlammer worm infected a SCADA system that controlled the Davis-Besse nuclear plant in Ohio. The worm shutdown the HMI of the supervisor SCADA systems that handled the plant's safety systems. This can be considered as Denial of Service (DoS) attack on ICS [4].
- Operation Ghoul: In August 2016, Kaspersky Labs unearthed a spear phishing campaign that was targeting industrial organizations. The attack started with an email that appeared to be coming from bank in UAE. This email is attached with a malware named HawkEye which collects the personal information through key strokes and clipboard data. Around 130 organizations across the globe were impacted with this attack. This attack can be considered as probing attack on ICS [5].
- New York Dam attack: U.S. Infrastructure online was attacked and infiltrated the computerized controls of a New York Dam. Justice department claimed it that it was done by an Iranian hacker. The attackers broke into the command and control

system of the dam in 2013, through a cellular modem. Even though the attack happened in 2013, it was only in 2016 that the cyber-attack was affirmed [6].

In 1990s, ICS cyber security awareness was very low and its perceived importance is even lower. Generally, it was viewed as a corporate Information Technology (IT) issue with little direct impact on powerplant or grid operation. Moreover, it was viewed as a hinderance to ICS technology advancements. From a security perspective, ICS were generally isolated networks and the concept of "security by obscurity" was alive and well. As security was not a consideration, there was little reason to question the need for tighter system integration.

The fundamental reason for securing ICS is to maintain the mission of the systems be it produce or deliver electricity, make or distribute gasoline, provide clean water etc. It is not simply possible to secure ICS electronically. However, it possible to increase the security measures and also minimize the possibility of unintentional incidents that have already costed hundreds of millions of euros as well as number of lives.

Traditionally, cyber security is in general viewed in the context of business IT systems and defense computer systems. Previously, ICS are frequently not viewed as "computers" nor they often consider to be susceptible to cyber threats. Consequently, cyber security is taught within the computer science departments focusing on traditional IT concepts. ICS generally do not utilize commercial-off-the-shelf operating systems and their computing resources are constrained. They often use proprietary real-time operating systems (RTOS) or embedded processors. These systems have different operating requirements and can be impacted by cyber vulnerabilities typical of IT systems and also cyber vulnerabilities unique to ICS. ICS continues to upgrade with advanced communication capabilities and are networked to improve process efficiency, productivity, regulatory compliance and safety. This networking can be within a facility or even between facilities that are continents apart. ICS are addressing various engineering disciplines such as control systems and applications but not in the computer security domain. When an ICS doesn't operate properly, it can result in impacts ranging from minor issues to catastrophic.

IT security in general deals with traditional commercial off-the-shelf hardware and software. There is always a possibility that the existing IT vulnerabilities such as probing can also be applicable to ICS domain. But these vulnerabilities can be of least priority in the ICS domain. The concept of security triad: Confidentiality, Integrity and Availability (CIA) defines the need for securing the systems (Fig. 1). In the IT domain, cyber-attacks often focus on acquisition of proprietary information and hence the CIA triad results in confidentiality being the most important attribute. However, in ICS domain the attacks tend to focus on destabilization of assets and moreover most of the attacks effect the integrity and the functionality of the ICS. This makes the integrity and availability as more important parameters than confidentially. Hence the research on securing ICS must focus on the addressing the integrity and availability as top priority followed by confidentiality.

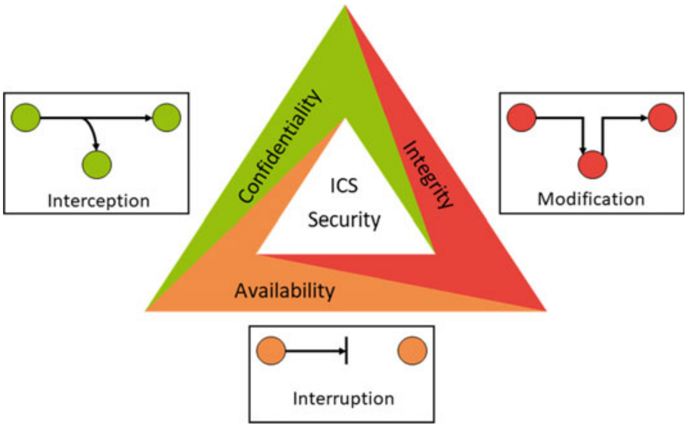


Fig. 1 CIA triad and their effects on the data flow

- **Confidentiality**: Is the property, that information is not made available or disclosed to unauthorized individuals, entities or processes. Interception of data leads to violation of confidentiality.
- **Integrity**: Is the property of maintaining and assuring the accuracy and completeness of data over its entire life-cycle or protecting information from being modified by unauthorized parties. External modification of the data leads to violation of integrity.
- **Availability**: Is the property to ensure that authorized parties or entities able to access the information or devices when needed. Interruption of data arrival or transfer leads to violation of availability.

From the Table 1 it is clear that the security measures followed by ICS are different from IT and requires special attention. The ICS design criteria was performance and safety and not security in the initial stage and the attacks which violates these criteria's needs to be identified.

While strong concerns about security of ICS, particularly in the context of critical national infrastructure, were expressed even in early 2000s [7]. The possible attacks on ICS can be mainly classified into two types namely network attacks, injection attacks.

Table 1 Priorities of security parameters in IT versus ICS

Priority	Information technology (IT)	Industrial Control Systems (ICS)
1	Confidentiality	Availability
2	Integrity	Integrity
3	Availability	Confidentiality

1.1 Network Attacks

ICS is a generic term which encompasses several types of control systems and associated technologies. Networked Control System (NCS) is a specific area in control systems where communication networks are used to close the control loop. The ICS architecture in Fig. 2 can be also considered as an example for NCS. The attacks types can change significantly from one year to the next. Many of the actual attacks involve combinations of vulnerabilities. Some common attacks on ICS can be:

– **Denial of Service (DoS)**: An attack against ICS to stop the proper functioning of some portion of an ICS or to effectively disable the entire system. These attacks can target on the connected physical system or the ICS itself. DoS against physical system vary from opening or closing of valves manually and switching to destruction of portions of the physical process that prevent operation. DoS against the ICS target the communication links or attempt to disable programs running on system endpoints which control the system, log data and govern communications [8]. DoS attack mainly effects '***availably***' which is main priority in ICS.
– **Probe**: It is an action taken or an object used for the purpose of learning something about the state of the network. It collects or monitors network activity and attempt to gain access to a computer and its files. Probe attack mainly effects the '***confidentiality***' in ICS.

A more detailed information about different types of network attacks and their detection mechanisms can be found in [9].

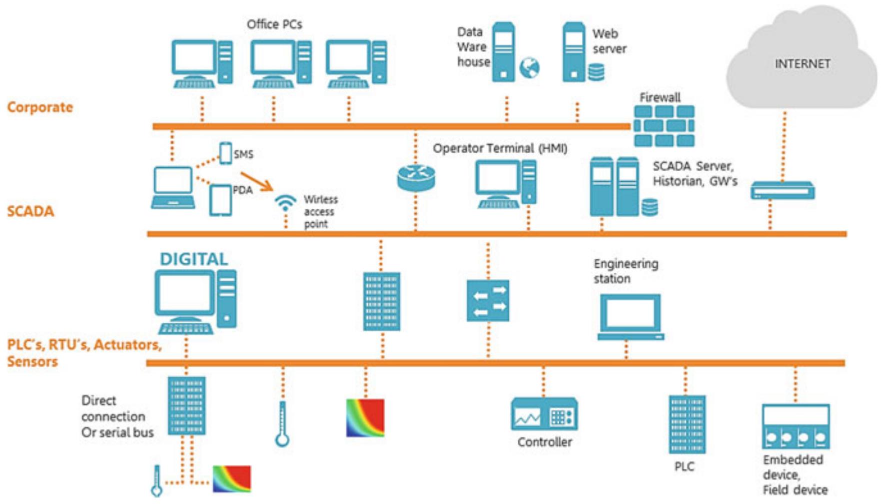


Fig. 2 Typical ICS architecture

1.2 Injection Attacks

Injection attacks also known as False Data Injection Attacks (FDIA) where attacker gains access to a critical process or process parameters in ICS and forces the system to execute newly introduced code or command. Injection attacks effect the '***integrity***' of an ICS. Some common injection attacks are:

- **Response Injection**: ICS protocols often take the first response packet to a query and reject subsequent responses as erroneous. This enables to craft response packets and use timing attacks to inject the responses into a network when they are expected by a client.
- **Measurement Injection**: Falsified process measurements are injected into the ICS in this type of attacks. The attacker simulates a process measurement such as a water level or gas pressure increasing or decreasing which in turn generate false actuations.
- **Command Injection**: False control or configuration commands are injected into the ICS in this type of attacks. Potential impacts of this attack include interruption of process control, interruption of device communications, unauthorized modification of device configurations etc.

FDIA affects the data integrity of packets by modifying their payloads. These are also considered as Intrusion via Remote Access from list of attacks mentioned and are considered as difficult to detect. From this we can say that the FDIA attacks were subtler than DoS while they are hard to detect and have not been thoroughly investigated especially in modern industries.

A more detailed information about the theory of injection attacks can be found at [10]. Different injection attacks and their detection mechanisms can be found in [8, 11]. From this we can say that any violation of the security parameter may lead to serious consequence and to avoid such violations a proper detection mechanism is necessary.

2 State of the Art

FDIA in short termed as injection attacks are previously popular in smart grid or power system applications. Most of the existing literature in identification of FDIA is only related to power systems or smart grids. But the application of Machine Learning (ML) and deep learning is application specific and research on securing ICS from FDIA is significant. A literature review on different existing techniques for identification of FDIA in ICS is discussed.

Securing modern industrial infrastructure is a key task and significant amount of research effort is being done to analyse, detect and handle failures. Reference [12] address the effects of different attacks on industrial control systems. A report from Federal Office of Information Security, Germany reveals Top 10 threats and counter measures in 2016 [13].

Several failure detection algorithms in dynamic systems were reviewed by the author in [14]. A Complete Survey on existing attacks and detection methods for false data injection are given in [15]. Different types of FDIA attacks such as maximum magnitude-based attack, wave based attack, positive or negative deviation attack and mixed attacks were discussed in [16]. The impact of FDIA in control systems is discussed in [10]. All the mentioned research concentrates on the impact of FDIA on ICS.

Since FDIA would result in abnormal behaviour, widely researched Anomaly Detection (AD) techniques can be applied for ICS. Due to the novel attack identification capabilities, ML and deep learning based AD techniques were exploited in many domains [17]. AD based network intrusion detection systems are most commonly used techniques to identify anomalous network patterns. Many techniques such as [18, 19] uses ML and deep learning techniques to identify the novel network attack patterns using deep learning techniques.

Anomaly based detection to detect strong attacks that feature the injection of large amount of spurious measurement data in very short time was provided by [20] in the smart grid applications. Three types of injection attacks were discussed in [21] who uses a network level water control system to provide a closed loop defence framework to secure cyber physical systems. Single-input, single output scheme is used to verify the performance of controlled auto-regressive moving average models is discussed in [22].

AD techniques were also used for monitoring sensors networks and abnormal event detection. Reference [23] mentioned the importance of anomaly detection-based approach and used knowledge database to identify the attacks. This approach becomes complex if the amount of sensor data to be watched is enormous and requires a constant update of knowledge base which is practically not possible. A Bayesian network based approach for anomaly detection was performed in [24]. They combined Bayesian networks with Kalman Filter for predicting sensor failures but they did not consider any possible attacks on the network. Several ML algorithms were also used to identify the network attacks. Supervised ML techniques such a feed forward neural networks [25] and unsupervised learning techniques such as self-organizing maps [26] were used in identifying the attacks in network traffic. The detection accuracies of the unsupervised learning techniques were not comparable with the supervised learning mechanism when the labels are available. A further analysis is necessary to understand the outcome of the unsupervised technique which needs through knowledge on the process.

A lot of study on FDIA on smart grids is available [27–29]. Some other techniques such as sate estimation [30], ML [31], sparse optimization [32] were also used for identification of attacks. Our research has identified that the impact of FDIA on ICS needs to be addressed more precisely and the research on identification of FDIA in industrial infrastructure is limited. A neural network based FDIA identification approach on automation plant was discussed in [33]. Due to limited data generation and complex simulation issues, we choose to implement our future concepts on the process control plant data and evaluate our results. In this paper, we use the processes control plant for data acquisition and preform AD for FDIA identification.

3 Proposed Approach

Attack identification is domain where it is not so easy to define a specific approach. A lot of pre-processing and feature extraction is necessary. Pre-processing techniques such as data reduction, data cleaning and data transformation are necessary to extract the exact information from the plant. Multiple feature extraction techniques need to be used in order to get meaningful information from the plant data. But out of huge list of features, proper features from statistical, mathematical and ML based needs to be selected based on the behaviour of the obtained sensors signals. If the labelling of obtained data is possible at least for training phase, deep learning techniques can classify the attacks in an efficient manner. Finally, the outcome of the deep learning AD strategy is classified into a normal or an attack class (Fig. 3).

3.1 Pre-processing

Pre-processing of data is a crucial step for various applications not only for deep learning. The effort on pre-processing depends on the characteristics of acquired data. It involves transforming raw data into an understandable format. The data obtained from real-world is often incomplete, inconsistent, lack certain common behaviour or trend and is likely to contain errors and corrupted values. Pre-processing techniques such as data cleaning, data transformation and data reduction are proven techniques to resolve the above-mentioned issues. Data cleaning detects and corrects corrupt or inaccurate records in the collected data. This includes removing of corrupted data which may occur due to equipment malfunction, handling of noisy data and outlier removal. Data transformation includes tasks such as smoothing, normalization, aggregation and generalization of acquired data. Normalization is the key task in data transformation which scales the data to a specified range. Min-Max normalization and z-score normalization techniques are most commonly used normalization techniques. Data reduction is usually done when acquired data is too big to handle or work with. Dimensionality reduction and aggregation are two common approaches

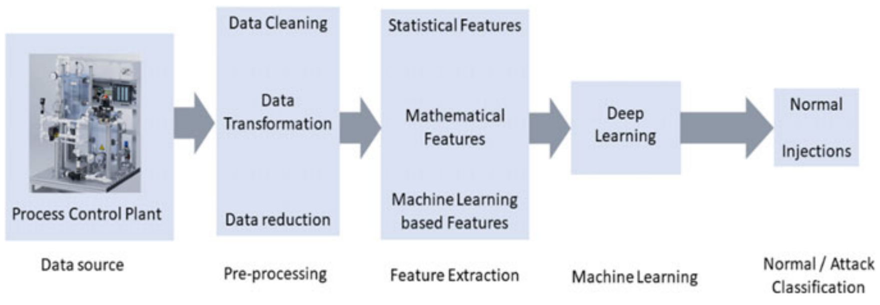


Fig. 3 Proposed architecture for injection attacks identification

in data reduction. A more detailed information about the data pre-processing can be found in [34]. Pre-processing prepares the data for further processing such as feature extraction.

3.2 Feature Extraction

Feature extraction is a key step in many ML applications such as pattern recognition and image processing. The derived features out of raw data intends to be more informative and non-redundant facilitating the subsequent learning and generalization steps and in some cases, leading to better human interpretations. Sometimes feature extraction is also considered as a data reduction mechanism as discussed in pre-processing. Different features and features extraction techniques are available. Some common and significant feature categories are statistical features, mathematical features and features extracted through ML techniques.

Statistical Features
Statistics, termed as a branch of mathematics dealing with the collection, analysis, interpretation and presentation of masses of numeric data. Statistical features are those which are defined and calculated through statistical analysis. Statistical analysis theory is the frequently used method for feature extraction of data in the time domain [35]. It can analyse according to the statistical laws when several objects and several indices are interrelated. Statistical methods are based on forceful theory that have lots of algorithms and can effectively analyse and process the data. Several statistical factors exist out of which some most commonly used are mean, median, variance, standard deviation, root means square etc. A huge list of statistical features can be found in [36].

Mathematical Features
Mathematical methods are applied on the raw or pre-processed data to obtain the meaningful information. Mathematical features are the most commonly extracted features on both time series and time independent transformations. Several mathematical functions from transformation theory can be used to translate the signal into a different domain. List of mathematical features include derivate, probability and stochastic process, estimation theory, numerical methods etc.

Machine Learning Based Features
ML techniques are not only used to perform classification or clustering but can also be used for feature extraction from raw data and complex data structures. The features such as efficient data compression and data reduction are performed with ML techniques for feature extraction. One good example for ML based feature extraction is Principal Component Analysis (PCA). Even deep learning techniques such as autoencoders and Boltzmann machines are popular for feature extraction. ML techniques are also used to extract features out of features i.e. the extracted features from other techniques or even from ML techniques can be again given to ML based feature extraction to get much refined or complex features.

The set of features extracted from raw data differs from the network packets and the choice of features extracted such as statistical, mathematical or ML based needs to be made based on the application and acquired data.

3.3 Deep Learning

Deep learning is a ML technique combining both supervised and unsupervised techniques inspired from human brain. Human brain has got multiple levels of representations with simple features at lower levels and more abstract features at higher levels. Similarly, deep learning consists of multiple hidden layers with initial layers representing the information at lower levels and the final layers representing the information in an abstract format. Some common deep learning architectures include Stacked Auto-Encoders (SAE), Deep Belief Networks (DBN), Convolutional Neural Networks (CNN) etc. CNN's are mostly used in image processing applications and requires huge dataset and training time. But the implementation of CNN for AD is novel.

Convolutional-Neural Networks
Convolutional Neural Networks (CNN) is one category of deep learning algorithm and are considered as an extension to the traditional feed forward neural networks. CNN have proven very effective in many application domains such as image recognition and classification, speech processing applications etc. Its effectiveness has been successfully proven in tasks such as identifying faces, objects and traffic sign detection mainly used in robotics and self-driving cars.

Four main operations of CNN comprise of Convolution layer, non-linear activation function such as Rectified Linear Unit (ReLU), pooling layer, and fully connected layer (classification).

– **Convolution Layer**: As the name itself indicates, the CNN got its name from convolution operations. The main task of convolution is to extract features from the input image. Convolution operation preserves the spatial relationship between pixels by learning image features using filter or a kernel. The output image out of convolution operation is termed as 'Activation Map' or 'Convolved Feature' or 'Featured Map'. The values of the filter or Kernel are updated automatically during the training process of CNN to learn the features of an image in a better way. The size of feature map is controlled by depth (corresponding to number of filters we use), stride (Number of pixels by which we slide the filter) and zero padding (padding images with zeros at the border). If the image is padded with zeros at the border then it is termed as wide convolution and if not, it is considered as a narrow convolution. More detailed information on convolution layer is discussed in [37]. The process of convolution operation and feature extraction from an image is show in Fig. 4.
– **Nonlinear Activation ReLU**: After every convolution operation, before generating the feature map, additional nonlinear function such as ReLU is being used

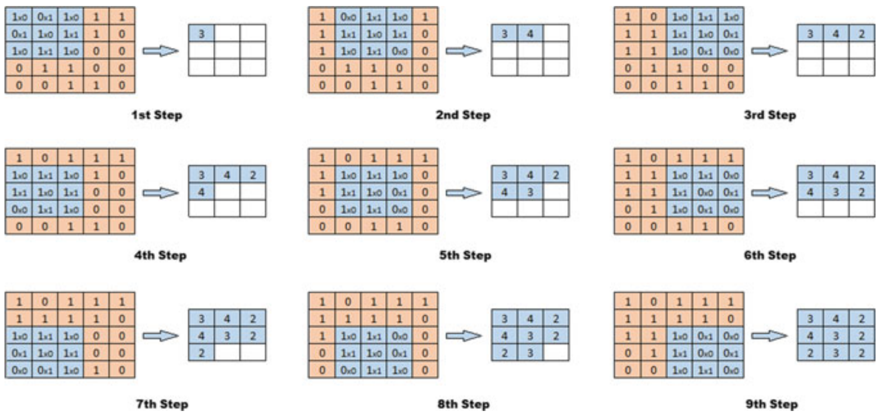


Fig. 4 Convolution process for feature extraction from an image

in CNN. ReLU stands for Rectified Linear Unit and is a non-linear operation. It is an element wise operation and replaces all negative pixel values in the feature map with zero. ReLU introduces the non-linear behavior to the CNN and traditional convolution operation is linear. Other non-linear activation functions such as tanh and sigmoid can also be used instead of ReLU. More detailed information on ReLU activation and other activation functions are discussed in [38].

$$f(x) = \max(0, x) \tag{1}$$

– **Pooling Layer**: Spatial pooling also termed as subsampling or down-sampling reduces the dimensionality of each feature map but retains the most important information out of the feature map. Spatial pooling can be of different types such as Max, Average, Sum etc. Max pooling has shown to work better in many applications. More detailed information on pooling layer is discussed in [39]. An example of max pooling is shown in Fig. 5. Where the operation is done through a filter size of 2×2 along with a stride value of two.
– **Fully Connected Layer**: This is a traditional multi-layer perceptron that uses a softmax activation function in the output layer. The term fully connected implies that every neuron in the previous layer is connected to every neuron in the next layer. The output of the convolution and pooling layers represent the high-level features of the input image. The fully connected layer uses these features for classifying the input image into various classes based on the training dataset. More detailed information on fully connected layer is discussed in [40].

Combining the above-mentioned key parameters forms the CNN. The convolution and pooling layers act as a feature extraction mechanism out of an image while the fully connected layer act as a classifier. More detailed discussion on CNN is discussed in [41]. Figure 6 will give a detailed overview of the above-mentioned concepts in

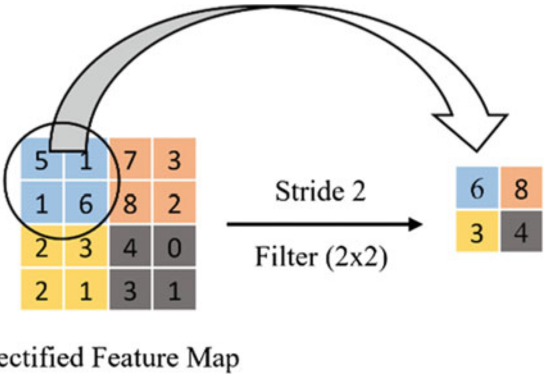


Fig. 5 Subsampling from rectified feature map

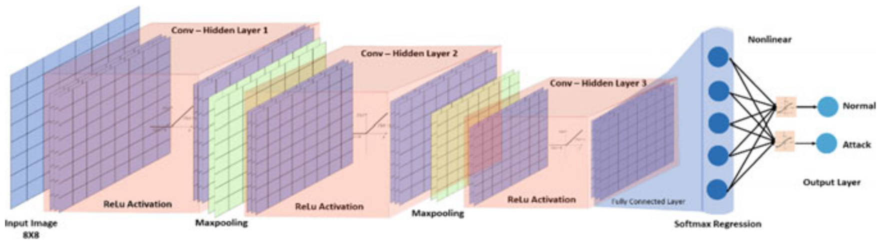


Fig. 6 CNN architecture for proposed mechanism

relation to our application of CNN for attack detection. The detailed functionality of the implemented CNN model is discussed later.

4 Dataset

To analyse the functionality of the developed model, a deep learning based security strategy is developed with a process control use case. The chair of "Integrated Automation" at Otto-von-Guericke University Magdeburg has a Process Control Plant in the Automation Laboratory. This plant was built by Festo Didactic and termed as MPS PA compact workstation with level, flow rate, pressure and temperature controlled system [42]. The following Fig. 7 gives an overview about the picture of the plant and the P&ID diagram of the plant.

Using a corresponding controller such as Samson Trovsi 6495, the level and flowrate-controlled system can be set up as a cascade control system. The level of each tank is measured via ultrasound sensors and the flow rate between the tanks are measured via flow sensors. Pressure sensor and temperature sensor also provided but they are not used for this experiment. A two-way ball valve with a pneumatic process

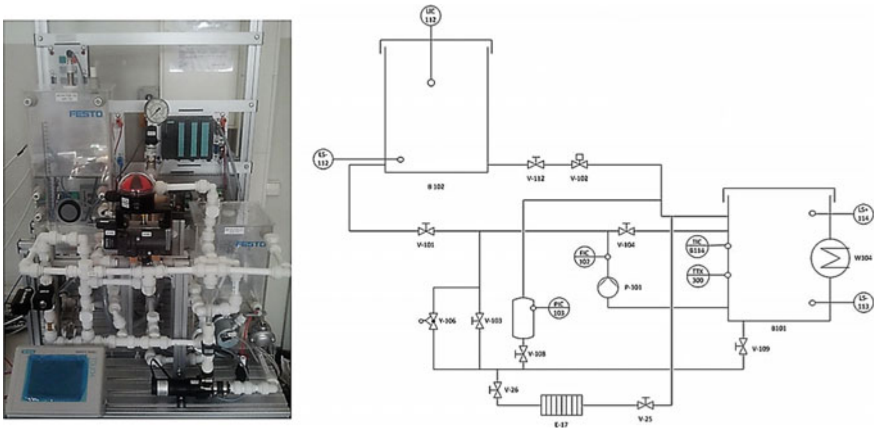


Fig. 7 Process control plant and P&ID diagram of the plant

actuator which connects the elevated tank and lower reservoir is used to control the manipulated variable. The pump is controlled via speed adjustment. The level sensor values as well as the pump actuation values from the plant are used for the evaluation. Simulated measurement injection attacks were injected in the level sensor data as well as command injection attacks were injected in pump data.

The attacks were simulated very effectively that they are hard to identify in general. For example, Fig. 8a represents the filtered tank level data and Fig. 8b represents the attack injected signal. We can see a specific difference between the two signals but it is hard to identify by just looking at Fig. 8b that there are some attacks injected. The injected attacks will be in normal range of the signal but are inserted in a specific pattern such that the values are complete opposite to the normal behaviour but within the normal range and only through specific features the difference is identified. Below figures shows the filtered signal and attacks inserted signal for both measurement injection and command injection (Figs. 8 and 9).

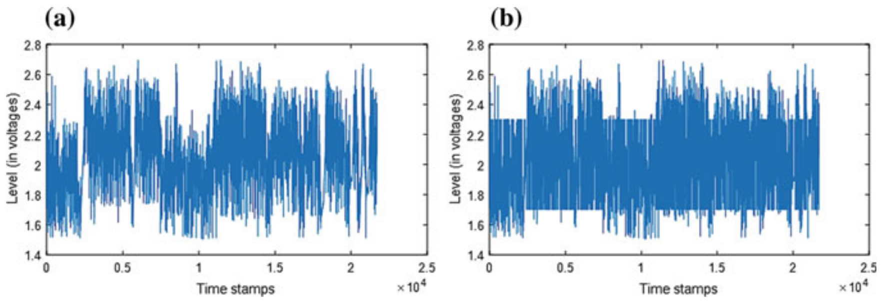


Fig. 8 Tank level sensor data **a** Filtered signal **b** Attack Injected Signal

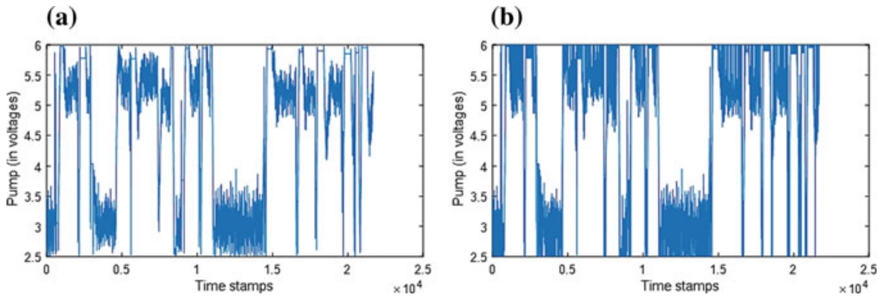


Fig. 9 Pump data **a** Filtered signal **b** Attack injected signal

The level sensor data is the simulated data from the level sensors of twin tank system as it is mentioned before. The level sensors are used to read the liquid level in the tank constantly for proper function of the two-tank system but this system behaviour can be manipulated with fault measurement injection to the sensor data by the attackers. Suppose, fault measurement data has been injected to the level sensor data then the scenario could be like this: due to fault measurement injection the sensor would read different level of liquid than the actual and caused the malfunctioning of the system. This malfunctioning could bring a great damage to the system and environment surrounding the system. After obtaining the sensors signals (with injections) the proposed architecture is implemented on the plant data and CNN is applied for attack classification.

5 Implementation

The flow diagram of implementing deep learning based AD for identifying FDIA in ICS is depicted in Fig. 10.

5.1 *Pre-Processing*

As the data obtained from raw sensors is very basic, it is hard to understand the attack patterns from the sensor data alone. Hence, some basic features through statistical and mathematical methods. In total 11 features are extracted for both level sensor and pump data. The obtained features are represented in following Figs. 11, 12, 13, 14, 15 and 16.

After extracting the respective features out of the sensor data, the features need to be normalized for better learning. The data has been normalized using Eq. 2 and after on the normalized value has been discretized into 20 intervals using a similar

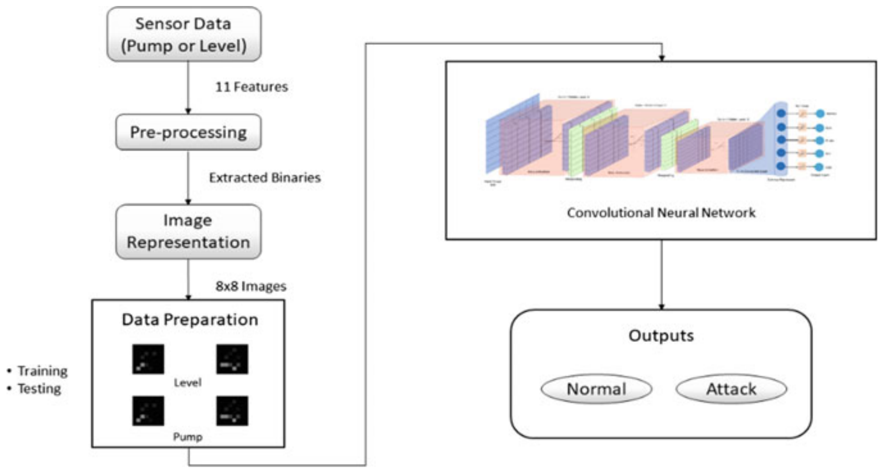


Fig. 10 Flow diagram of implemented CNN model from proposed architecture

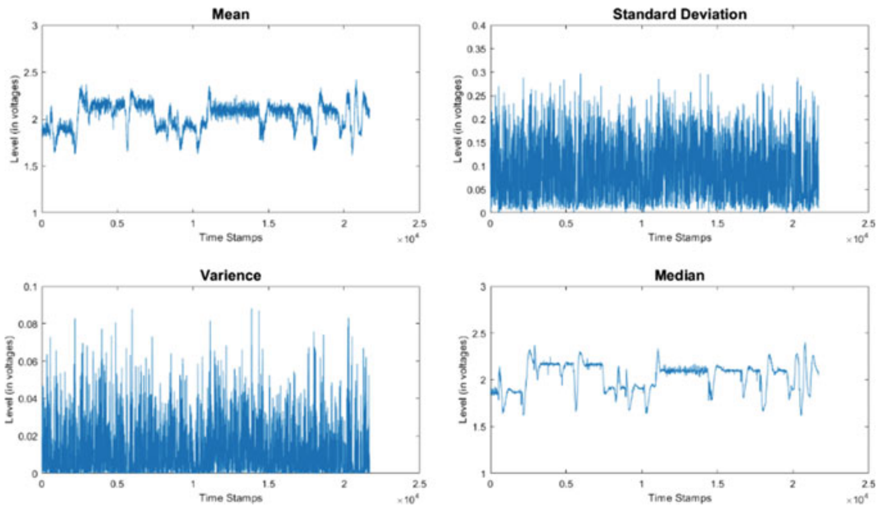


Fig. 11 Extracted level sensor features (Mean, Std, Varience & Median)

discretizer's like Fig. 17. The used discretizer in the figure has a standard scaler of value 0.1 but here it has been used of value 0.05. Hence, it has divided into 20 intervals instead of 10. These 20 intervals have been assigned after on with 20 different bits for binary extraction.

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2}$$

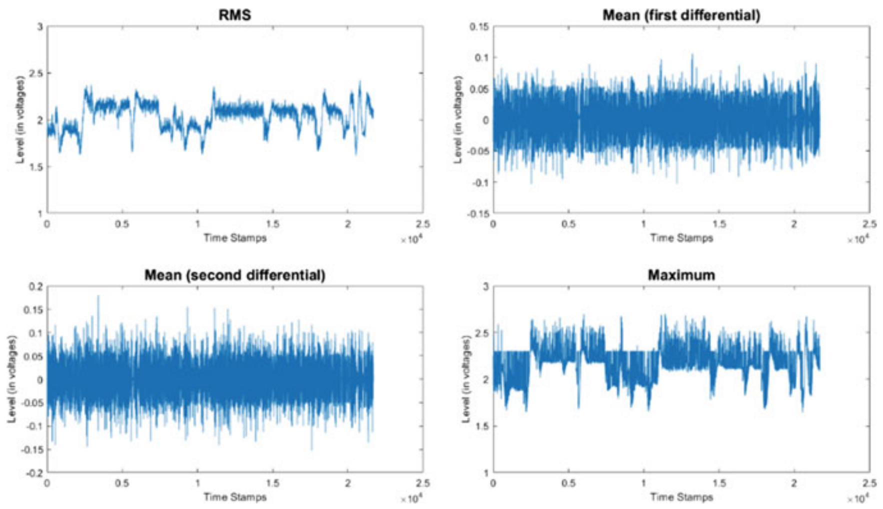


Fig. 12 Extracted level sensor features (RMS, Mean-1st & 2nd differential and Max)

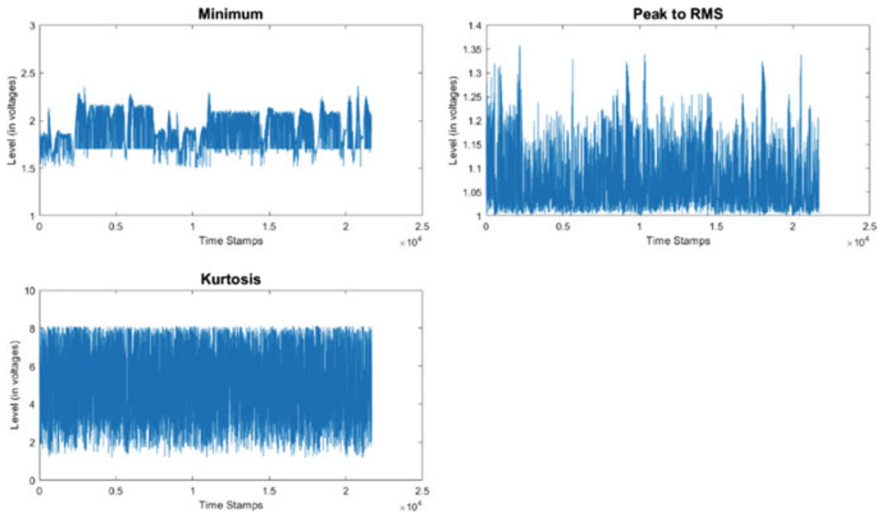


Fig. 13 Extracted level sensor features (Min, Peak to RMS & Kurtosis)

The first four listed features above are shown in Fig. 11.

In Fig. 12 the extracted features using RMS, first differential mean, second differential mean and maximum value are shown below.

The extracted features namely minimum, peak to RMS and kurtosis are represented in Fig. 13.

The enlisted features above are also extracted from the pump sensor data and they are represented in the following Figures.

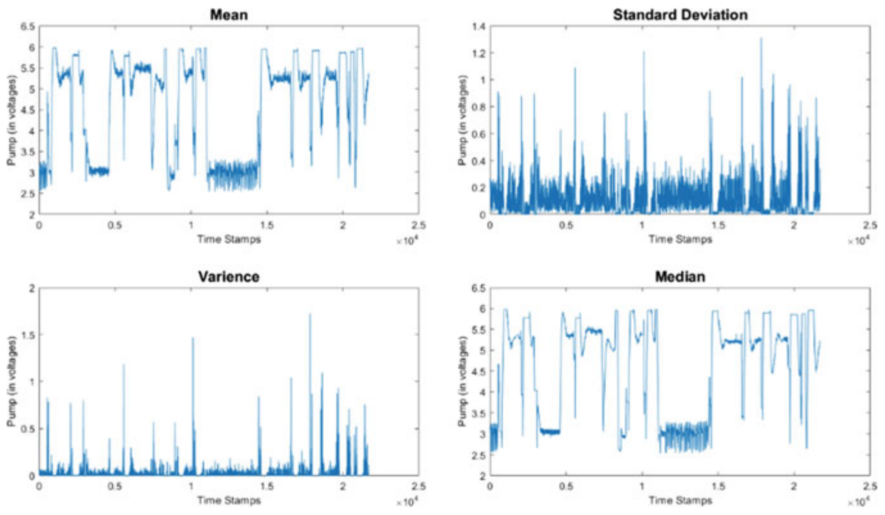


Fig. 14 Extracted pump sensor features (Mean, Std, Varience & Median)

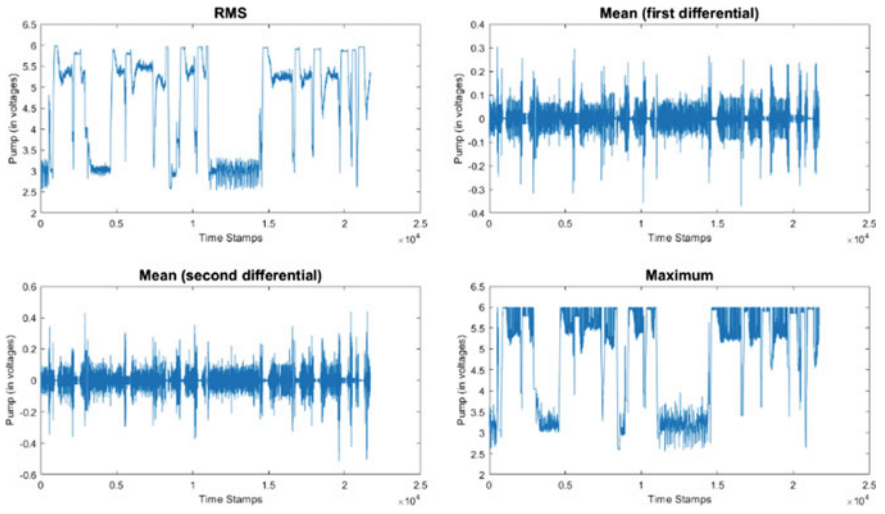


Fig. 15 Extracted pump sensor features (RMS, Mean-1st & 2nd differential and Max)

The first four listed features of pump sensor data are shown in Fig. 14.

In Fig. 15 the extracted features using RMS, first differential mean, second differential mean and maximum value are shown below.

The remaining three extracted features of pump sensor data is shown in Fig. 16.

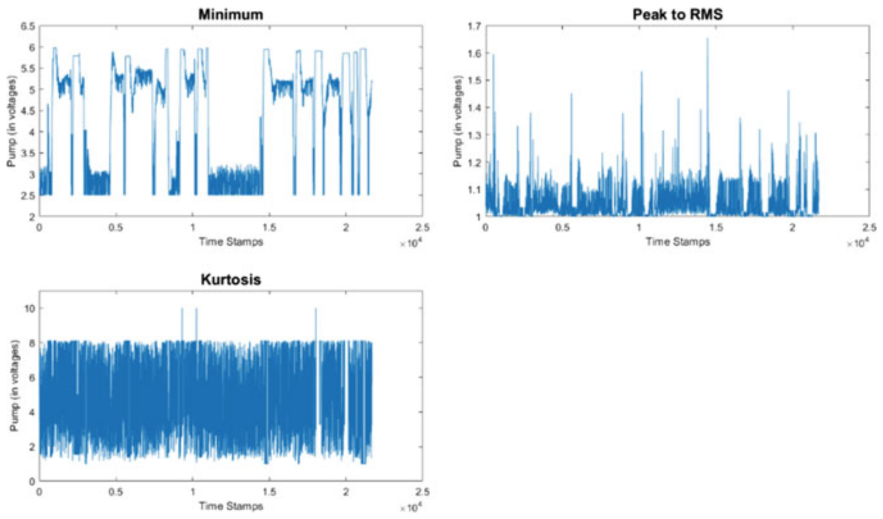


Fig. 16 Extracted pump sensor features (Min, Peak to RMS & Kurtosis)

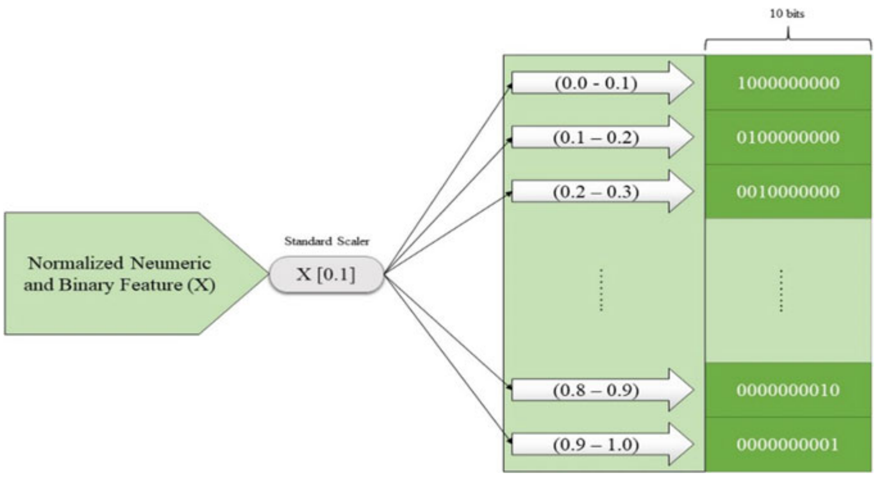


Fig. 17 A discretise mechanism for binary extraction

5.2 *Image Representation*

The simulated sensor data has been converted into 240 bits binary vector by assigning 20 bits to each of 11 extracted features via the discritizer. Now, each 8 bits of 240 bits have been converted into grayscale pixels and then the pixels are reshaped into 6×6 matrix where, the vacant six pixels are filled with zeros. In order to create 8×8 pixel matrix the generated pixel matrix is padded with zeros circularly as shown in

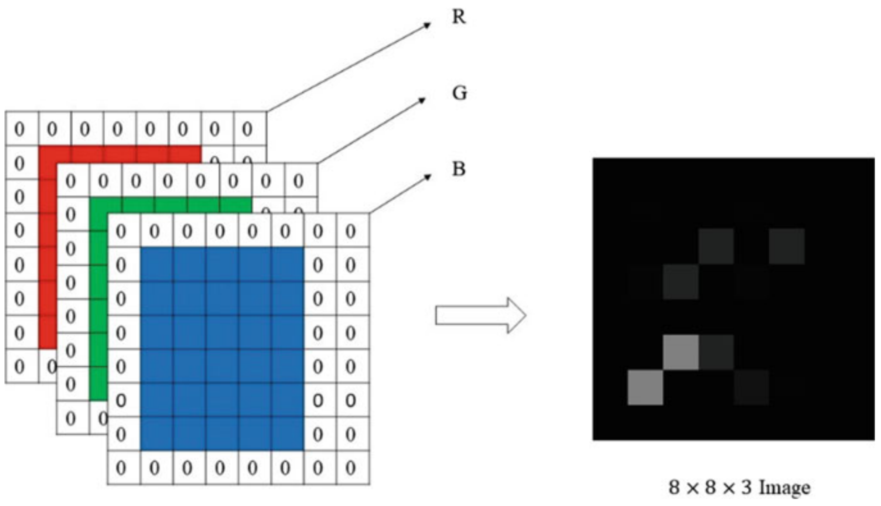


Fig. 18 Image representation of sensor data

Fig. 18. Finally, the same 8 × 8 pixel matrix has been sent to the red, green and blue channels to represent converted sensor data as RGB images and prepared them for training and testing purposes.

5.3 Training of CNN

The Matlab Neural Network (NN) toolbox has been updated recently for deep learning algorithms [43]. The Matlab release of 2017 [44] and beyond have offered modern deep neural networks models especially for image classification or recognition namely: facial recognition, motion detection, autonomous driving, pedestrian detection and autonomous parking. The NN toolbox could be used to create and interconnect the layers of deep NN by using Matlab commands. For example, Matlab has been offered few pre-trained NNs for deep learning which has made it easier to use deep learning algorithms, it could be used having primary knowledge on NN and computer vision algorithms.

Some basic parameters needs to be configured while training the CNN algorithm are follows:

– **Input Image Size (M N C)**: It is the size of the input image where, M is the height of the image, N is the width of the image an C is the number of channels such as: for grayscale image $C = 1$ and for color image $C = 3$.
– **Filter_Size**: It is declaration of the size of filters to use. For instance, if the value is assigned as 3 then it indicates that the filter height and width are 3.
– **Number_of_Filters**: It is the number of filters to use for convolution operation.

- **Pooling_Size**: It is the size of the rectangular pooling map. For example, if the value is assigned as 2 then the pooling map would be a square of size 2.
- **Stride_Value**: ‘Stride’ is a name value pair argument and the step size of filtering is defined by it.
- **Number_of_Classes**: It is the number of input category Such as for intrusion detection generally there are two attack category namely normal and attack therefore, the number of classes must be 2 in this case.

The configured CNN model is then trained for the purpose of attack classification. Where, ‘sgdm’ stochastic gradient decent method is used for learning approach. The learning rate has been set as ‘piecewise’ i.e. with initial learning rate 0.01 and would decrease by a factor of 0.1. ‘MaxEpochs is set to 20’ which represents the number of times the whole training data is sent for training iterations. The training dataset is split to training as well as validation. ‘Validation Frequency is set to 300’ such that after 300 iterations the validation of the trained network is made, the value after that many times of iterations the results would be validated. The validation patience has been set to 5 for reducing the training cost. It means, if the validation accuracy is not changed for five validation periods in a row then the training will be stopped resulting in model not getting overfitted (Fig. 19).

Pertained networks like Inception-v3, ResNet-50, ResNet-101, GoogLeNet, AlexNet, VGG-16, and VGG-19 are available. These networks can be used by customizing their network layers according to the need. Furthermore, it also offers training the network on both CPU and GPU environments in Matlab.

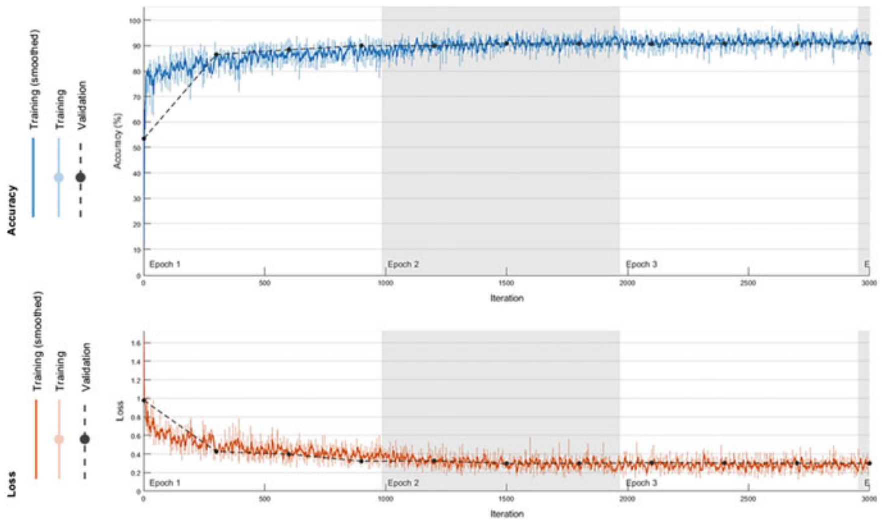


Fig. 19 Training validation curve of trained CNN model

6 Results and Discussion

The evaluation of the results of the trained model are done based on the following parameters represented in Fig. 20. The performance assessment for both the training and testing of proposed models are done using the parameters True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The presented confusion matrix in the Fig. 20 is for two class model.

The performance metrics used to assess the performance of the proposed model are:

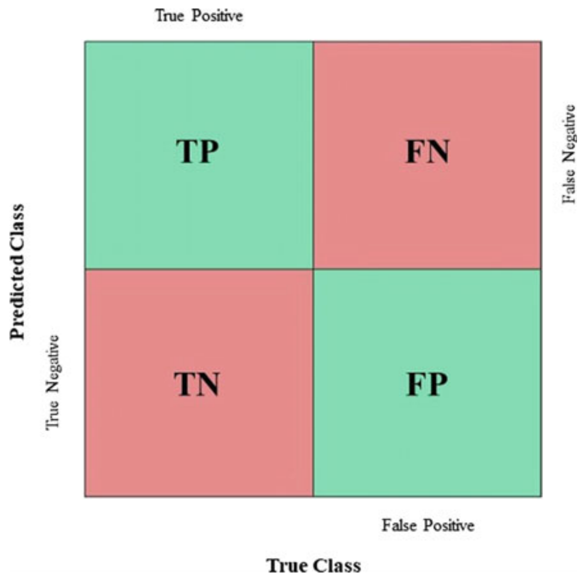
- Accuracy: The number of detected samples correctly over the total number of samples, which is represented in percentages.
- Precision (***P***): The precision is mentioned for Positive Predictive Value (PPV) and it is calculated in percentage via dividing the TP by the summation of TP and FP as given in Eq. 3.

$$P = \frac{TP}{TP + FP} \times 100 \tag{3}$$

- Recall (***R***): The recall is mentioned as the TP rate and it is calculated in percentage via TP by the summation of TP and FN as given in Eq. 4.

$$R = \frac{TP}{TP + FN} \times 100 \tag{4}$$

Fig. 20 Parameters of confusion matrix



- F-Measure: The F-Measure is a measurement for representing the test accuracy and mentioned as harmonic mean of values P and R and can be calculated by using the Eq. 5.

$$F = \frac{2 \times P \times R}{P + R} \tag{5}$$

- Cohen's kappa coefficient (k): The Cohen's kappa coefficient is a statistical measure of inter-rater reliability or the agreement that is used to assess qualitative items. representing the test accuracy and mentioned as harmonic mean of values P and R and can be calculated by using the Eq. 6.

$$k = \frac{p_o - p_e}{1 - p_e} \tag{6}$$

where p_o is the relative observed agreement among ratters (similar to accuracy) and p_e is the hypothetical probability of chance agreement, using observed data to calculate the probabilities of each observer randomly seeing each category.

$$p_0 = \frac{TP + FP}{TP + FN + TN + FP} \tag{7}$$

$$p_e = p_{yes} + p_{no} \tag{8}$$

$$p_{yes} = \frac{TP + FN}{TP + FN + TN + FP} \tag{9}$$

$$p_{no} = \frac{TN + FP}{TP + FN + TN + FP} \tag{10}$$

The outcome of the performance metrics evaluation is given in Tables 2, 3, 4, 5, 6, 7 and 8.

Table 2 CNN accuracy for attack identification on level sensor data

Type of framework	Classes		Overall detection rate (%)
	Normal (%)	Attack (%)	
CNN	97.50	94.90	96.70

Table 3 Performance metrics evaluation of CNN model on attack identification on level sensor data

Parameter	Normal (%)	Attack (%)
Precision	97.74	94.37
Recall	97.52	94.86
F-Measure	97.63	94.61
Cohen's kappa coefficient	0.96	0.93

Table 4 CNN accuracy for attack identification on pump data

Type of framework	Classes		Overall detection rate (%)
	Normal (%)	Attack (%)	
CNN	95.90	92.70	94.90

Table 5 Performamnce metrics evaluation of CNN model on attack identification on pump data

Parameter	Normal (%)	Attack (%)
Precision	96.90	90.40
Recall	95.87	92.69
F-Measure	96.38	91.53
Cohen's kappa coefficient	0.94	0.90

Table 6 Training and testing times of CNN on GeForce GTX960M GPU with 4 GB on board memory

	Train time (s)	Training samples	Test time (s)	Testing samples
Level sensor data	184.32	17289	2.12	4400
Pump data	278.14	17289	2.28	4400

Table 7 Performance comparison of CNN with other deep learning approaches of attack identification on level sensor data

Type of framework	Classes		Overall detection rate (%)
	Normal (%)	Attack (%)	
CNN	97.50	94.90	96.70
SAE	97.44	96.69	97.18
DBN	86.39	88.51	87.11

7 Conclusion and Future Works

This paper mainly concerns about the development of deep learning-based security strategy for securing ICS against FDIA. For this purpose, CNN is chosen as a deep learning algorithm for classification of normal and attack classes. Entire implementation was implemented in MATLAB. A process control plant from Institute for Automation Engineering is used to generate the data. These data are injected with

Table 8 Performance comparison of CNN with other deep learning approaches of attack identification on pump data

Type of framework	Classes		Overall detection rate (%)
	Normal (%)	Attack (%)	
CNN	95.90	92.70	94.90
SAE	97.07	94.11	96.09
DBN	97.45	57.05	84.13

FDIA in a strategical manner. From raw sensor data, some statistical and mathematical features are extracted. These features are later converted to image format to train the CNN model. Different parameters of CNN model are discussed and our choice of those parameters is justified. Based on the validation parameters and patience in training the network avoids the proposed model from overfitting. Finally, the performance of the proposed CNN model is evaluated with the metrics such as accuracy, precision, F-measure, recall and Cohen's kappa coefficient. The results look promising and ensures the capabilities of CNN for using them as a security strategy against cyber threats on ICS. The performance results of the CNN are compared with our previous results with other deep learning approaches such as SAE and DBN. Obviously, CNN outperformance DBN in terms of detecting both normal and attack but competes almost equivalently with SAE. But this is purely dependent on the application and generated dataset. After observing the training time on CPU which is more than 12 h, the entire training process of CNN is shifted onto the GPU and the training time shows that the there is a significant acceleration of the training process on the GPU.

In future, huge datasets and more attack types are necessary to evaluate the performance more in detail. A combination of multiple feature extraction techniques especially ML based feature extraction is under progress. A combination of different deep learning techniques and ML techniques is under plan to improve the efficiency of the security mechanism. Use of other deep learning approaches such as Recurrent Neural Networks (RNN) are in progress.

References

1. ISA: ISA99, Industrial automation and control systems security. https://www.isa.org/isa99/. Accessed 07 Mar 2019
2. The White House and Washington: PRESIDENTIAL DECISION DIRECTIVE/NSC-63. https://fas.org/irp/offdocs/pdd/pdd-63.htm. Accessed 07 Mar 2019
3. Nigam, R.: (Known) SCADA attacks over the year, Security Research. https://blog.fortinet.com/2015/02/12/known-scada-attacks-over-the-years (2015). Accessed 15 Sept 2017
4. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: The spread of the sapphire/slammer worm
5. Legezo, D.: Operation Ghoul: learning from the targeted attack analysis to protect your business. https://www.kaspersky.com/blog/ghoul/5897/ (2016). Accessed 15 Sept 2017
6. Thompson, M.: Iranian cyber attack on New York dam shows future of war. http://time.com/4270728/iran-cyber-attack-dam-fbi/ (2016). Accessed 15 Sept 2017
7. Colbert, E.J., Kott, A.: Cyber-Security of SCADA and other Industrial Control Systems, 63rd edn. Springer (2016)
8. Morris, T., Gao, W.: Industrial control system cyber attacks. In: International Symposium on ICS SCADA Cyber Security Research, pp. 22–29 (2013)
9. Mangrulkar, N.S.: Network attacks and their detection mechanisms: A Review **90**(9), 36–39 (2014)
10. Mo, Y., Sinopoli, B.: False data injection attacks in control systems. In: Conference on DecisionControl (2010)
11. Potluri, S., Diedrich, C., Sangala, G.K.R.: Identifying false data injection attacks in industrial control systems using artificial neural networks. In: Emergeing Technology in Factory Automation ETFA 2017 (2017)

12. Huang, H., Kasiviswanathan, S., Electric, G.: Streaming anomaly detection using online matrix sketching **9**(3), 1–15 (2015)
13. F. O. for I. Security. Industrial control system security (2016)
14. Willsky, A.S.: A survey of design methods for failure detection in dynamic systems. Automatica **12**(6), 601–611 (1976)
15. Yu, Z.H., Chin, W.L.: Blind false data injection attack using PCA approximation method in smart grid. IEEE Trans. Smart Grid **6**(3), 1219–1226 (2015)
16. Kamesh, Sakthi Priya, N.: Security enhancement of authenticated RFID generation. Int. J. Appl. Eng. Res. **9**(22), 5968–5974 (2014)
17. Alrawashdeh, K., Purdy, C.: Toward an online anomaly intrusion detection system based on deep learning. In: 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 195–200 (2016)
18. Kaur, H., Minhas, J., Singh, G.: A review of machine learning based anomaly detection techniques. Int. J. Comput. Appl. Technol. Res. **2**(2), 185–187 (2013)
19. Van, N.T., Thinh, T.N.: An anomaly-based network intrusion detection system using deep learning. In: International Conference on System Science and Engineering (ICSSE) (2017)
20. Yu, W., Griffith D., Ge, L., Bhattarai, S., Golmie, N.: An integrated detection system against false data injection attacks in the smart grid. Secur. Commun. Netw. **8**, 91–109 (2014)
21. Huang, S., Zhou, C., Yang, S., Qin, Y.: Cyber-physical system security for networked **12**, 567–578 (2015)
22. Pang, Z., Hou, F., Zhou, Y.: Design of false data injection attacks for output tracking control of CARMA systems. In: International Conference on Information and Automation, pp. 1273–1277 (2015)
23. Rabatel, J., Bringay, S., Poncelet, P.: Anomaly detection in monitoring sensor data for preventive maintenance. Expert Syst. Appl. **38**(6), 7003–7015 (2011)
24. Hill, D.J., Minsker, B.S., Amir, E.: Real-time Bayesian anomaly detection in streaming environmental data. Water Resour. Res. **46**(4), 1–16 (2010)
25. Pradhan, S.K.S.M., Pradhanm, S.K.: Anomaly detection using artificial neural networks. Int. J. Eng. Sci. Emerg. Technol. **2**(1), 29–36 (2012)
26. Siripanadorn, S.: Anomaly detection using self-organizing map and wavelets in wireless sensor networks. In: Proceedings of the 10th WSEA, pp. 291–297 (2010)
27. Guan, Z., Sun, N., Xu, Y.: A comprehensive survey of false data injection in smart grid. Mob. Comput. **8**(1) (2015)
28. Wang, D., Guan, X., Liu, T., Gu, Y., Sun, Y., Liu, Y.: A survey on bad data injection attack in smart grid
29. Baig, Z.A., Amoudi, A.: An analysis of smart grid attacks and countermeasures **8**(8) (2013)
30. Anwar, A.: Vulnerabilities of smart grid state estimation against false data injection attack cyber incidents in different sector in renewable energy integration, green energy and technology (2014)
31. Esmalifalak, M., Member, S., Liu, L., Member, S.: Detecting stealthy false data injection using machine learning in smart grid, 1–9 (2014)
32. Hao, J., Member, S., Piechocki, R.J., Kaleshi, D.: Sparse malicious false data injection attacks and defense mechanisms in smart grids **3203**, 1–12 (2015)
33. Potluri, S., Diedrich, C., Sangala, G.K.R.: Identifying false data injection attacks in industrial control systems using artificial neural networks. In: 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–8 (2017)
34. Famili, A., Shen, W., Weber, R., Simoudis, E.: Data preprocessing and intelligent data analysis **1,** 3–23 (1997)
35. Siekmann, J., Wahlster, W.: Advanced intelligent computing theories and applications
36. MaxStat: tools for scientific data analysis—Statistics. http://www.maxstat.de/statistical-tests.html. Accessed 15 Sep 2017
37. Wu, J.: Introduction to convolutional neural networks. 1–28, (2016)
38. Agarap, A.F.: Deep learning using rectified linear units (ReLU), 1 (2018)

39. Wu, H., Gu, X.: Max-pooling dropout for regularization of convolutional neural networks. Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence, Lecture Notes in Bioinformatics), vol. 9489, pp. 46–54 (2015)
40. ujjwalkarn: An intuitive explanation of convolutional neural networks. The data science blog. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (2016). Accessed 06 May 2018
41. Bhandare, A., Bhide, M., Gokhale, P., Chandavarkar, R.: Applications of convolutional neural networks. Int. J. Comput. Sci. Inf. Technol. **7**(5), 2206–2215 (2016)
42. Festo Didactic: MPS® PA Compact-Workstation mit Füllstands-, Durchfluss-, Druck- und Temperaturregelstrecken. http://www.festo-didactic.com/de-de/lernsysteme/prozessautomation,regelungstechnik/compact-workstation/mps-pa-compact-workstation-mit-fuellstands-,durchfluss-,druck-und-temperaturregelstrecken.htm?fbid=ZGUuZGUuNTQ0LjEzLjE4Ljg4Mi40Mzc2. Accessed 15 Sep 2017
43. MathWorks: Training a model from Scratch—MATLAB & Simulink. https://www.mathworks.com/solutions/deep-learning/examples/training-a-model-from-scratch.html. Accessed 07 Mar 2019
44. MathWorks: Options for training deep learning neural network. https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html. Accessed 07 Mar 2019

Deep Learning for Wireless Communications

Tugba Erpek, Timothy J. O'Shea, Yalin E. Sagduyu, Yi Shi and T. Charles Clancy

Abstract Existing communication systems exhibit inherent limitations in translating theory to practice when handling the complexity of optimization for emerging wireless applications with high degrees of freedom. Deep learning has a strong potential to overcome this challenge via data-driven solutions and improve the performance of wireless systems in utilizing limited spectrum resources. In this chapter, we first describe how deep learning is used to design an end-to-end communication system using autoencoders. This flexible design effectively captures channel impairments and optimizes transmitter and receiver operations jointly in single-antenna, multiple-antenna, and multiuser communications. Next, we present the benefits of deep learning in spectrum situation awareness ranging from channel modeling and estimation to signal detection and classification tasks. Deep learning improves the performance when the model-based methods fail. Finally, we discuss how deep learning applies to wireless communication security. In this context, adversarial machine learning provides novel means to launch and defend against wireless attacks. These

T. Erpek (✉) · T. J. O'Shea · T. C. Clancy
Virginia Tech, Arlington, VA, USA
e-mail: terpek@vt.edu

T. J. O'Shea
e-mail: oshea@vt.edu

T. C. Clancy
e-mail: tcc@vt.edu

T. J. O'Shea
DeepSig, Inc., Arlington, VA, USA

Y. E. Sagduyu · Y. Shi
Intelligent Automation, Inc., Rockville, MD, USA
e-mail: ysagduyu@i-a-i.com

Y. Shi
e-mail: yshi@vt.edu

Y. Shi
Virginia Tech, Blacksburg, VA, USA

© Springer Nature Switzerland AG 2020
W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_9

applications demonstrate the power of deep learning in providing novel means to design, optimize, adapt, and secure wireless communications.

Keywords Deep learning · Wireless systems · Physical layer · End-to-end communication · Signal detection and classification · Wireless security

1 Introduction

It is of paramount importance to deliver information in wireless medium from one point to another quickly, reliably, and securely. Wireless communications is a field of rich expert knowledge that involves designing waveforms (e.g., long-term evolution (LTE) and fifth generation mobile communications systems (5G)), modeling channels (e.g., multipath fading), handling interference (e.g., jamming) and traffic (e.g., network congestion) effects, compensating for radio hardware imperfections (e.g., RF front end non-linearity), developing communication chains (i.e., transmitter and receiver), recovering distorted symbols and bits (e.g., forward error correction), and supporting wireless security (e.g., jammer detection). The design and implementation of conventional communication systems are built upon strong probabilistic analytic models and assumptions. However, existing communication theories exhibit strong limitations in utilizing limited spectrum resources and handling the complexity of optimization for emerging wireless applications (such as spectrum sharing, multimedia, Internet of Things (IoT), virtual and augmented reality), each with high degrees of freedom. Instead of following a rigid design, new generations of wireless systems empowered by cognitive radio [1] can learn from spectrum data, and optimize their spectrum utilization to enhance their performance. These smart communication systems rely on various detection, classification, and prediction tasks such as signal detection and signal type identification in spectrum sensing to increase situational awareness. To achieve the tasks set forth in this vision, machine learning (especially deep learning) provides powerful automated means for communication systems to learn from spectrum data and adapt to spectrum dynamics [2].

Wireless communications combine various waveform, channel, traffic, and interference effects, each with its own complex structures that quickly change over time, as illustrated in Fig. 1. The data underlying wireless communications come in large

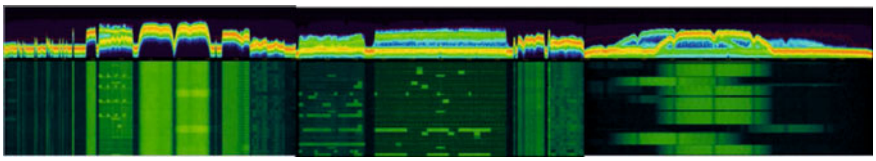


Fig. 1 Example spectrum data plots: 900/1800 MHz cellular and 2.4 GHz industrial, scientific and medical (ISM) radio bands

volumes and at high rates, e.g., gigabits per second in 5G, and is subject to harsh interference and various security threats due to the shared nature of wireless medium. Traditional modeling and machine learning techniques often fall short of capturing the delicate relationship between highly complex spectrum data and communication design, while deep learning has emerged as a viable means to meet data rate, speed, reliability, and security requirements of wireless communication systems. One motivating example in this regard is from signal classification where a receiver needs to classify the received signals [3] based on waveform features, e.g., modulation used at the transmitter that adds the information to the carrier signal by varying its properties (e.g., amplitude, frequency, or phase). This signal classification task is essential in dynamic spectrum access (DSA) where a transmitter (secondary user) needs to first identify signals of primary users (such as TV broadcast networks) who has the license to operate on that frequency and then avoid interference with them (by not transmitting at the same time on the same frequency). It was shown that deep learning based on convolutional neural networks (CNN) achieves significantly higher accuracy in signal classification compared to feature-based classifiers where signal features such as the bandwidth or second-order statistics are used to discriminate different signals [3].

This chapter presents methodologies and algorithms to apply deep learning to wireless communications in three main areas.

1. Deep learning to design *end-to-end (physical layer) communication chain* (Sect. 2).
2. Deep learning to support *spectrum situation awareness* (Sect. 3).
3. Deep learning for *wireless security* to launch and defend wireless attacks (Sect. 4).

In Sect. 2, we formulate an end-to-end physical layer communications chain (transmitter and receiver) as an *autoencoder* that is based on two *deep neural networks* (DNNs), namely an *encoder* for the transmitter functionalities such as modulation and coding, and a *decoder* for the receiver functionalities such as demodulation and decoding. By incorporating the channel impairments in the design process of autoencoder, we demonstrate the performance gains over conventional communication schemes. In Sect. 3, we present how to use different DNNs such as *feedforward*, *convolutional*, and *recurrent neural networks* for a variety of spectrum awareness applications ranging from channel modeling and estimation to spectrum sensing and signal classification. To support fast response to spectrum changes, we discuss the use of *autoencoder* to extract latent features from wireless communications data and the use of *generative adversarial networks* (GANs) for spectrum data augmentation to shorten spectrum sensing period. Due to the open and broadcast nature of wireless medium, wireless communications are prone to various attacks such as jamming. In Sect. 4, we present emerging techniques built upon *adversarial deep learning* to gain new insights on how to attack wireless communication systems more intelligently compared to conventional wireless attack such as jamming data transmissions. We also discuss a defense mechanism where the adversary can be fooled when adversarial deep learning is applied by the wireless system itself.

2 Deep Learning for End-to-End Communication Chain

The fundamental problem of communication systems is to transmit a message such as a bit stream from a transmitter using radio waves and reproduce it either exactly or approximately at a receiver [4]. The focus in this section is on the physical layer of the Open Systems Interconnection (OSI) model. Conventional communication systems split signal processing into a chain of multiple independent blocks separately at the transmitter and receiver, and optimize each block individually for a different functionality. Figure 2 shows the block diagram of a conventional communication system. The source encoder compresses the input data and removes redundancy. Channel encoder adds redundancy on the output of the source encoder in a controlled way to cope with the negative effects of the communication medium. Modulator block changes the signal characteristics based on the desired data rate and received signal level at the receiver (if adaptive modulation is used at the transmitter). The communication channel distorts and attenuates the transmitted signal. Furthermore, noise is added to the signal at the receiver due to the receiver hardware impairments. Each communication block at the transmitter prepares the signal to the negative effects of the communication medium and receiver noise while still trying to maximize the system efficiency. These operations are reversed at the receiver in the same order to reconstruct the information sent by the transmitter. This approach has led to efficient, versatile, and controllable communication systems that we have today with individually optimized processing blocks. However, this individual optimization process does not necessarily optimize the overall communication system. For example, the separation of source and channel coding (at the physical layer) is known to be suboptimal [5]. The benefit in joint design of communication blocks is not limited to physical layer but spans other layers such as medium access control at link layer and routing at network layer [6]. Motivated by this flexible design paradigm, deep learning provides automated means to treat multiple communications blocks at the transmitter and the receiver jointly by training them as combinations of DNNs.

MIMO systems improve spectral efficiency by using multiple antennas at both transmitter and receiver to increase the communication range and data rate. Different signals are transmitted from each antenna at the same frequency. Then each antenna at the receiver receives superposition (namely, interference) of the signals from transmitter antennas in addition to the channel impairments (also observed for single antenna systems). The traditional algorithms developed for MIMO signal detection are iterative reconstruction approaches and their computational complexity is impractical for many fast-paced applications that require effective and fast signal processing to provide high data rates [7, 8]. Model-driven MIMO detection techniques can be applied to optimize the trainable parameters with deep learning and improve the detection performance. As an example, a MIMO detector was built in [7] by unfolding a projected gradient descent method. The deep learning architecture used a compressed sufficient statistic as an input in this scheme. Another model-driven deep learning network was used in [9] for the orthogonal approximate message passing algorithm.

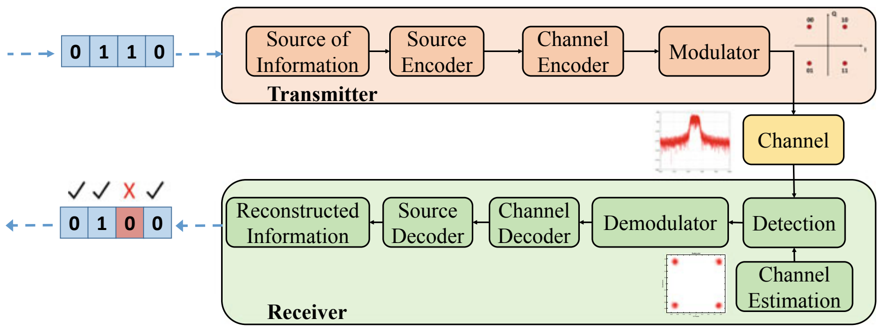


Fig. 2 Conventional communication system block diagram

Multiuser communication systems, where multiple transmitters and/or receivers communicate at the same time on the same frequency, allow efficient use of the spectrum, e.g., in an *interference channel* (IC), multiple transmitters communicate with their intended receivers on the same channel. The signals received from unintended transmitters introduce additional interference which needs to be eliminated with precoding at the transmitters and signal processing at the receivers. The capacity region for IC in *weak*, *strong* and *very strong interference regimes* has been studied extensively [10–12]. Non-orthogonal multiple access (NOMA) has emerged to improve the spectral efficiency by allowing some degree of interference at receivers that can be efficiently controlled across interference regimes [13]. However, the computational complexity of such capacity-achieving schemes is typically high to be realized in practical systems.

Recently, deep learning-based end-to-end communication systems have been developed for single antenna [14, 15], multiple antenna [16], and multiuser [14, 17] systems to improve the performance of the traditional approaches by jointly optimizing the transmitter and the receiver as an *autoencoder* instead of optimizing individual modules both at the transmitter and receiver. Autoencoder is a DNN that consists of an encoder that learns a (latent) representation of the given data and a decoder that reconstructs the input data from the encoded data [18]. In this setting, joint modulation and coding at the transmitter corresponds to the encoder, and joint decoding and demodulation at the receiver corresponds to the decoder. The joint optimization includes multiple transmitter and receivers for the multiuser case to learn and eliminate the additional interference caused by multiple transmitters. The following sections will present the autoencoder-based communication system implementations and their performance evaluation.

2.1 Single Antenna Systems

A communication system consists of a transmitter, a receiver, and channel that carries information from the transmitter to the receiver. A fundamental new way to think about communication system design is to formulate it as an end-to-end reconstruction task that seeks to jointly optimize transmitter and receiver components in a single process using autoencoders [14]. As in the conventional communication systems, the transmitter wants to communicate one out of M possible messages $s \in \mathcal{M} = \{1, 2, \ldots, M\}$ to the receiver making n discrete uses of the channel. It applies the modulation process $f : \mathcal{M} \mapsto \mathbb{R}^n$ to the message s to generate the transmitted signal $\mathbf{x} = f(s) \in \mathbb{R}^n$. The input symbols from a discrete alphabet are mapped to the points (complex numbers) on the constellation diagram as part of digital modulation. The digital modulation schemes for conventional communication systems have pre-defined constellation diagrams. The symbols are constructed by grouping the input bits based on the desired data rate. The desired data rate determines the constellation scheme to be used. Figure 3 shows the constellation diagrams for the binary phase shift keying (BPSK), quadrature phase shift keying (QPSK), and 16-quadrature amplitude modulation (QAM) as example of digital modulation schemes and their symbol mapping. Linear decision regions make the decoding task relatively simpler at the receiver. For the autoencoder system, the output constellation diagrams are not pre-defined. They are optimized based on the desired performance metric, i.e., the symbol error rate to be reduced at the receiver.

The hardware of the transmitter imposes an energy constraint $\|\mathbf{x}\|_2^2 \leq n$, amplitude constraint $|x_i| \leq 1\, \forall i$, or an average power constraint $\mathbb{E}\left[|x_i|^2\right] \leq 1\, \forall i$ on $\mathbf{x}$. The data rate of this communication system is calculated as $R = k/n$ [bit/channel use], where $k = \log_2(M)$ is the number of input bits and n can be considered as the output of a forward error correction scheme where it includes both the input bits and redundant bits to mitigate the channel effects. As a result, the notation (n,k) means that a communication system sends one out of $M = 2^k$ messages (i.e., k bits) through n channel uses. The communication channel is described by the conditional probability density function $p(\mathbf{y}|\mathbf{x})$, where $\mathbf{y} \in \mathbb{R}^n$ denotes the received signal. Upon reception

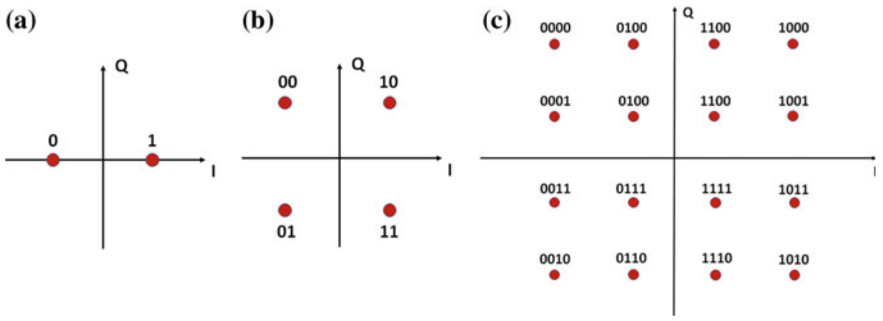


Fig. 3 Example of digital modulation constellations **a** BPSK, **b** QPSK, **c** 16-QAM

of $\mathbf{y}$, the receiver applies the transformation $g : \mathbb{R}^n \mapsto \mathcal{M}$ to produce the estimate $\hat{s}$ of the transmitted message s. Mapping $\mathbf{x}$ to $\mathbf{y}$ is optimized in a *channel autoencoder* so that the transmitted message can be recovered with a small probability of error. In other words, autoencoders used in many other deep learning application areas typically remove redundancy from input data by compressing it; however, the channel autoencoder adds controlled redundancy to learn an intermediate representation robust to channel perturbations.

The block diagram of the channel autoencoder scheme is shown in Fig. 4. The input symbol is represented as a one-hot vector. The transmitter consists of a feedforward neural network (FNN) with multiple dense layers. The output of the last dense layer is reshaped to have two values that represent complex numbers with real (in-phase, I) and imaginary (quadrature, Q) parts for each modulated input symbol. The normalization layer ensures that physical constraints on $\mathbf{x}$ are met. The channel is represented by an additive noise layer with a fixed variance $\beta = (2RE_b/N_0)^{-1}$, where E_b/N_0 denotes the energy per bit (E_b) to noise power spectral density (N_0) ratio. The receiver is also implemented as an FNN. Its last layer uses a softmax activation whose output $\mathbf{p} \in (0, 1)^M$ is a probability vector over all possible messages. The index of the element of $\mathbf{p}$ with the highest probability is selected as the decoded message. The autoencoder is trained using stochastic gradient descent (SGD) algorithm on the set of all possible messages $s \in \mathcal{M}$ using the well suited categorical cross-entropy loss function between $\mathbf{1}_s$ and $\mathbf{p}$. The noise value changes in every training instance. Noise layer is used in the forward pass to distort the transmitted signal. It is ignored in the backward pass.

Figure 5a compares the block error rate (BLER), i.e., $\Pr(\hat{s} \neq s)$, of a communication system employing BPSK modulation and a Hamming (7, 4) code with either binary hard-decision decoding or maximum likelihood decoding (MLD) against the BLER achieved by the trained autoencoder (7, 4) (with fixed energy constraint $\|\mathbf{x}\|_2^2 = n$). Autoencoder is trained at $E_b/N_0 = 7$ dB using Adam [19] optimizer with learning rate 0.001. Both systems operate at rate $R = 4/7$. The BLER of uncoded

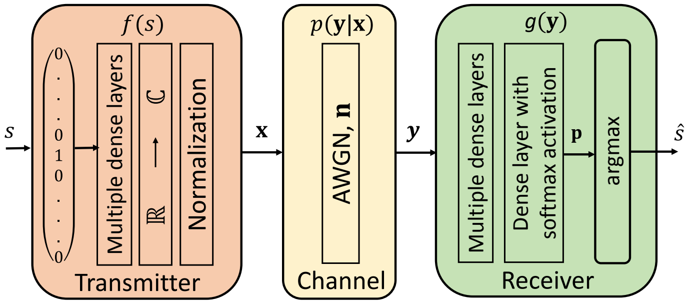


Fig. 4 A communication system over an additive white Gaussian noise (AWGN) channel represented as an autoencoder. The input s is encoded as a one-hot vector, the output is a probability distribution over all possible messages. The message with the highest probability is selected as output $\hat{s}$

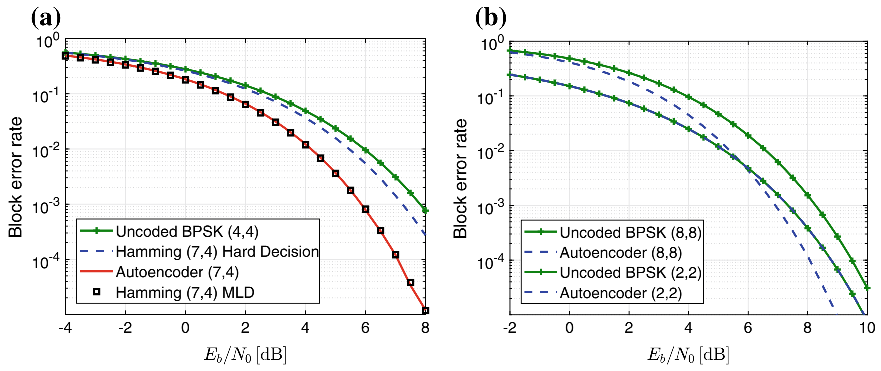


Fig. 5 BLER versus E_b/N_0 for the autoencoder and several baseline communication schemes [14]

BPSK $(4, 4)$ is also included for comparison. The autoencoder learns encoder and decoder functions without any prior knowledge that achieve the same performance as the Hamming $(7, 4)$ code with MLD. Table 1 shows the number of neural network layers used at the encoder (transmitter) and decoder (receiver) of the autoencoder system.

Figure 5b shows the performance curves for $(8, 8)$ and $(2, 2)$ communication systems when $R = 1$. The autoencoder achieves the same BLER as uncoded BPSK for $(2, 2)$ system and it outperforms the latter for $(8, 8)$ system, implying that it has learned a joint coding and modulation scheme, such that a coding gain is achieved. Figure 6 shows the constellations $\mathbf{x}$ of all messages for different values of (n, k) as complex constellation points, i.e., the x- and y-axes correspond to the first and second transmitted symbols, respectively. Figure 6a shows the simple $(2, 2)$ system that converges rapidly to a classical QPSK constellation (see Fig. 3b) with some arbitrary rotation. Similarly, Fig. 6b shows a $(4, 2)$ system that leads to a rotated 16-PSK constellation where each constellation point has the same amplitude. Once an average power normalization is used instead of a fixed energy constraint, the constellation plot results in a mixed pentagonal/hexagonal grid arrangement as shown in Fig. 6c. This diagram can be compared to the 16-QAM constellation as shown in Fig. 3c.

Table 1 Layout of the autoencoder used in Fig. 5a and b

Transmitter		Receiver	
Layer	Output dimensions	Layer	Output dimensions
Input	M	Input	n
Dense + ReLU	M	Dense + ReLU	M
Dense + linear	n	Dense + softmax	M

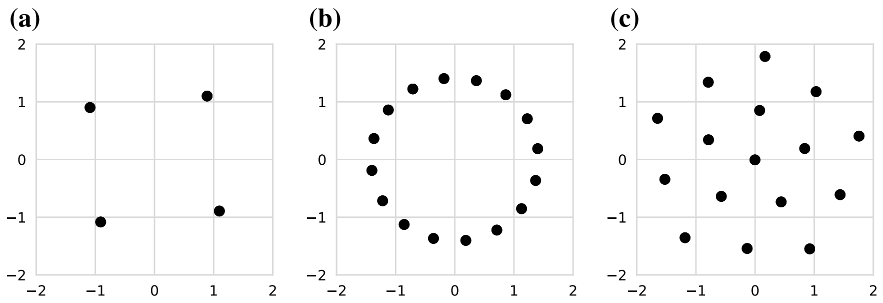


Fig. 6 Constellations produced by autoencoders using parameters (n, k): **a** (2, 2) **b** (2, 4), **c** (2, 4) with average power constraint

In addition to promising results for the channel autoencoder implementation with simulated channels, over-the-air transmissions have also verified the feasibility of building, training, and running a complete communication system solely composed of DNNs using unsynchronized off-the-shelf software-defined radios (SDRs) and open-source deep learning software libraries [15]. Hardware implementation introduces additional challenges to the system such as the unknown channel transfer function. The autoencoder concept works when there is a differentiable mathematical expression of the channel's transfer function for each training example. A two-step training strategy is used to overcome this issue where the autoencoder is first trained with a stochastic channel model that closely approximates the real channel model. During operation time, the receiver's DNN parameters are fine-tuned using *transfer learning* approach. A comparison of the BLER performance of the channel autoencoder system implemented on the SDR platform with that of a conventional communication scheme shows competitive performance close to 1 dB without extensive hyperparameter tuning [15].

Transfer learning approach still provides suboptimal performance for the channel autoencoder since the channel model used during the training differs from the one experienced during operation time. A training algorithm that iterates between the supervised training of the receiver and *reinforcement learning*-based training of the transmitter was developed in [20] for different channel models including AWGN and Rayleigh block-fading (RBF) channels.

2.2 Multiple Antenna Systems

MIMO wireless systems are widely used today in cellular and wireless local area network (LAN) communications. A MIMO system exploits multipath propagation through multiple antennas at the transmitter and receiver to achieve different types of gains including beamforming, spatial diversity, spatial multiplexing gains, and interference reduction. Spatial diversity is used to increase coverage and robustness by using space-time block codes (STBC) [21, 22]. Same information is precoded and transmitted in multiple time slots in this approach. Spatial multiplexing is used to increase the throughput by sending different symbols from each antenna element [23, 24]. In a closed-loop system, the receiver performs channel estimation and sends this channel state information (CSI) back to the transmitter. The CSI is used at the transmitter to precode the signal due to interference created by the additional antenna elements operating at the same frequency. The developed MIMO schemes for both spatial diversity and multiplexing rely on analytically obtained (typically fixed) precoding and decoding schemes.

Deep learning has been used for MIMO detection at the receivers to improve the performance using model-driven deep learning networks [7, 9, 25]. In Sect. 2.1, the channel autoencoder was used to train a communication system with a single antenna. The autoencoder concept is also applied to the MIMO systems where many MIMO tasks are combined into a single end-to-end encoding and decoding process which can be jointly optimized to minimize symbol error rate (SER) for specific channel conditions [16]. A MIMO autoencoder system with N_t antennas at the transmitter and N_r antennas at the receiver is shown in Fig. 7. Symbols $s_i, i = 1, \ldots, N_t$, are inputs to the communication system. Each symbol has k bits of information. By varying k, the data rate of the autoencoder system can be adjusted. The input symbols are combined and represented with a single integer in the range of $[0, 2^{kN_t})$ as an input to the encoder (transmitter) and are encoded to form N_t parallel complex transmit streams, $\mathbf{x_i}$, as output, where $i = 1, \ldots, N_t$. There are different channel models developed for MIMO systems such as [26]. A Rayleigh fading channel is used in this example which leads to a full rank channel matrix. In this case, full benefit is achieved from the MIMO system since the received signal paths for each antenna are uncorrelated. The signal received at the decoder (receiver) can be modeled as $\mathbf{y} = \mathbf{hx} + \mathbf{n}$ where $\mathbf{h}$ is an $N_r \times N_t$ channel matrix with circularly symmetric complex Gaussian entries of zero mean and unit variance, $\mathbf{x}$ is an $N_t \times 1$ vector with modulated symbols with an average power constraint of P such that $\mathbb{E}[\mathbf{x}^*\mathbf{x}] \leq P$ where $\mathbf{x}^*$ denotes the Hermitian of $\mathbf{x}$, and $\mathbf{n}$ is an $(N_r \times 1)$ vector which is the AWGN at the receiver with $\mathbb{E}[\mathbf{nn}^*] = \sigma^2 \mathbf{I}_{N_r \times N_r}$. Estimated symbols $\hat{s}_i$, where $i = 1, \ldots, N_r$, are the outputs. Every modulated symbol at the transmitter corresponds to a single discrete use of the channel and the communication rate of the system is $\min(N_t, N_r) \cdot k$ bits.

The autoencoder is trained using channel realizations drawn from Rayleigh distribution. The transmitter communicates one out of 2^k possible messages from each antenna. The transmitter is designed using an FNN architecture. The input symbols go through an embedding layer followed by dense layers. Embedding layer turns positive integers to dense vectors of fixed size. The output of the embedding layer is converted to a one-dimensional tensor before going in to the dense layers using a flatten layer. Batch normalization [27] is used after embedding layer and every dense layer. The output of the last dense layer is reshaped to generate complex numbers as the output; i.e., even indices as the real part and odd indices as the imaginary part.

The transmitter has an average power constraint. The *normalization* layer normalizes the transmitter output so that the average power constraint is satisfied; i.e., $\mathbb{E}[\mathbf{x}^*\mathbf{x}] \leq P$. As in the single antenna case, the transmitter output, $\mathbf{x}$ can be thought

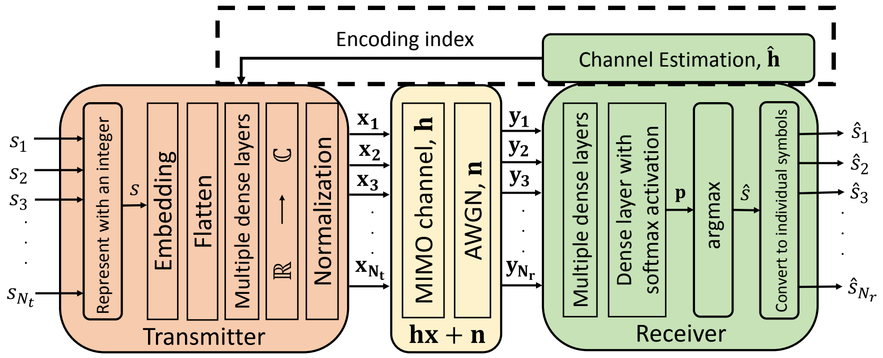


Fig. 7 MIMO channel autoencoder trained using a constant channel

as modulated symbols as in conventional communication systems. Instead of using a known constellation scheme with linear decision regions such as BPSK or QPSK, the optimal constellation points are learned by the autoencoder system over time.

A *multiplication* layer is built to perform complex multiplication, **hx**, and the *noise* layer introduces noise, **n**, to the autoencoder system. The input symbols and the noise change in every training instance and the noise variance σ is adjusted at both training and test time to simulate varying levels of signal-to-noise ratio (SNR).

The receiver is also designed using an FNN architecture. The symbols received at the receiver, $\mathbf{y_i}$, where $i = 1, \ldots, N_r$, go through multiple dense layers with the last layer with softmax activation that provides a probability for each symbol with a sum equal to 1. The codeword with the highest probability is selected as the output.

During training, the transmitter and receiver are optimized jointly to determine the weights and biases for both of the FNNs that minimize the reconstruction loss. There are total of 2^{kN_t} output classes. Categorical cross-entropy loss function (ℓ_{CE}) is used for optimization using gradient descent which is given by

$$\ell_{CE}(\boldsymbol{\theta}) = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=0}^{2^{kN_t}-1} p'_{o,j}\log(p_{o,j}), \tag{1}$$

where M is the mini-batch size, $\boldsymbol{\theta}$ is the set of neural network parameters, $p_{o,j}$ is the softmax layer's output probability for output class j for observation o, and $p'_{o,j}$ is the binary indicator (0 or 1) if class label j is the correct classification for observation o. Weight updates are computed based on the loss gradient using back-propagation algorithm with Adam [19] optimizer. In this case, a forward pass, $f(s, \boldsymbol{\theta})$, and a backward pass, $\frac{\partial \ell_{CE}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, are iteratively computed and a weight update is given by $\delta w = -\eta\frac{\partial \ell_{CE}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ with η representing the learning rate.

Channel estimation can be performed either using conventional or machine learning-based methods during test phase (channel estimation block in Fig. 7). **h** is the channel matrix and $\hat{\mathbf{h}}$ is the channel estimation at the receiver. During real-time operation, the receiver performs channel estimation and sends the index of the best encoding to the transmitter through the designated feedback channel. The cognitive transmitter will change the encoding scheme on-the-fly to minimize SER. As a result, a closed-loop system will be used during operation time as shown in Fig. 7.

Channel estimation error at the receiver leads to a sub-optimal encoding scheme to be selected both at the transmitter and the receiver, and translates to a performance loss. A minimum mean square error (MMSE) channel estimator is used at the receiver. Assuming $\mathbf{h} = \hat{\mathbf{h}} + \tilde{\mathbf{h}}$ where $\hat{\mathbf{h}}$ is the channel estimation matrix and $\tilde{\mathbf{h}}$ is the channel estimation error, the variance of $\tilde{\mathbf{h}}$ using an MMSE channel estimator is given as [28]:

$$\sigma_{\tilde{h}}^2 = \frac{1}{1 + \frac{\rho_\tau}{N_t} T_\tau} , \tag{2}$$

where ρ_τ is the SNR during the training phase and T_τ is the number of training samples. Equation (2) was used in [29, 30] for closed-loop MIMO systems with both channel estimation and feedback (from receiver to transmitter) to perform channel-guided precoding at the transmitter. Different error variances are introduced to the channels (originally used for training) in test time to measure the impact of channel estimation error.

A closed-loop MIMO system using singular value decomposition (SVD)-based precoding technique at the transmitter [23] is implemented as the baseline. The channel matrix, $\mathbf{h}$, can be written as $\mathbf{h} = \mathbf{U}\Lambda\mathbf{V}^*$ where $\mathbf{U}$ and $\mathbf{V}$ are $N_r \times N_r$ and $N_t \times N_t$ unitary matrices, respectively. Λ is a diagonal matrix with the singular values of $\mathbf{h}$. To eliminate the interference at each antenna, the channel is diagonalized by precoding the symbols at the transmitter and decoding at the receiver using the CSI. In this model, the received signal is written as $\tilde{\mathbf{y}} = \Lambda\tilde{\mathbf{x}} + \tilde{\mathbf{n}}$ where $\tilde{\mathbf{x}} = \mathbf{V}\mathbf{x}$, $\tilde{\mathbf{y}} = \mathbf{U}^*\mathbf{y}$ and $\tilde{\mathbf{n}} = \mathbf{U}^*\mathbf{n}$. The distribution of $\tilde{\mathbf{n}}$ is the same as $\mathbf{n}$ with $\tilde{\mathbf{n}} \sim \mathcal{N}(\mu, \sigma^2\mathbf{I}_{N_r})$.

The performance of a 2×2 autoencoder system is evaluated and compared with the baseline performance. The noise variance, σ^2, is set to 1 and $N_t = N_r$. A closed-loop system with perfect CSI (no channel estimation error) at the transmitter is assumed for the baseline simulation. QPSK modulation is used to modulate the input bits. Equal power is used at each antenna during transmission. A 2×2 autoencoder system is developed using 2 bits per symbol to match the bit rate with the baseline. Keras [31] with Tensorflow [32] backend is used for the autoencoder implementation based on deep learning. The FNN structures for the transmitter and receiver are shown in Table 2.

Figure 8a shows the SNR versus SER curves of the learned communication system compared to the baseline when no channel estimation error is assumed for both of the schemes. Promising results are obtained with the autoencoder approach when nonlinear constellation schemes are allowed at the transmitter. There is more than 10 dB gain at an SER of 10^{-2} when the autoencoder is used.

It is assumed that the transmitter and receiver will be trained for specific channel instances and resulting neural network parameters (weights and biases) will be stored in the memory. During operation time, the receiver will perform channel estimation

Table 2 FNN structures used at the transmitter and receiver

	Transmitter		Receiver	
Layers	# neurons	Activation function	# neurons	Activation function
Input	2		4	
1	32	ReLU	8	ReLU
2	16	ReLU	16	ReLU
3	8	ReLU	32	ReLU
Output	4	Linear	16	Softmax

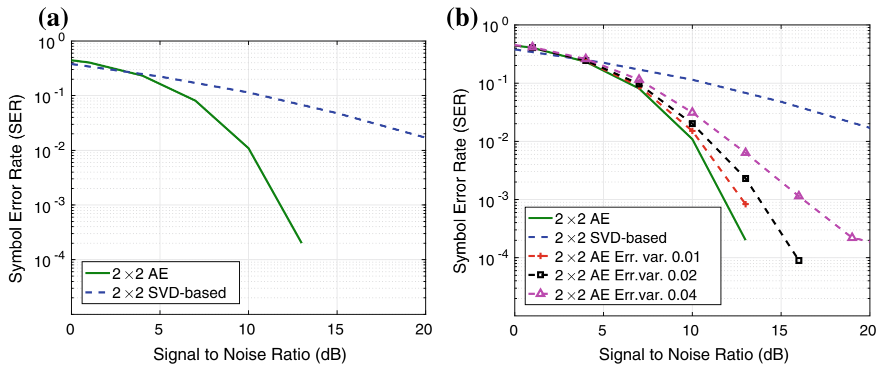


Fig. 8 **a** SER performance comparison of conventional and learned 2 × 2 spatial multiplexing schemes for a constant channel with perfect CSI, **b** The effect of channel estimation error on the performance of learned 2 × 2 spatial multiplexing scheme for constant channel

and send the index of the encodings that will be used to the transmitter. There will be channel estimation error at the receiver, which increases with decreasing number of training symbols [28]. Next, the performance of the developed autoencoder system when there is channel estimation error is analyzed using an MMSE channel estimator at the receiver. It is assumed that the training time increases with decreasing SNR and the system performance is analyzed when the channel estimation error variances are 0.01, 0.02 and 0.04. The autoencoder system is first trained with a given channel matrix, **h**. Then the output of the autoencoder architecture, weights, and biases are saved and the channel with the estimation error is provided during the operation time. Figure 8b shows the performance results. The autoencoder performance degrades with increasing channel estimation error, as expected. Error variance of 0.04 is the maximum that the system can tolerate.

2.3 Multiple User Systems

The autoencoder concept described in Sect. 2.1 was extended to multiple transmitters and receivers that operate at the same frequency for single antenna systems in [14] and for multiple antenna systems in [17]. A two-user AWGN interference channel was considered in [14] as shown in Fig. 9a.

Transmitter 1 wants to communicate message $s_1 \in \mathbb{M}$ to Receiver 1 and simultaneously, Transmitter 2 wants to communicate message $s_2 \in \mathbb{M}$ to Receiver 2. Extensions to K users with possibly different rates and other channel types are straightforward. Both transmitter-receiver pairs are implemented as FNNs. The encoder and decoder architectures are the same as described in Sect. 2.1. However, the transmitted messages interfere at the receivers in this case. The signal received at each receiver is given by

$$\mathbf{y}_1 = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{n}_1, \quad \mathbf{y}_2 = \mathbf{x}_2 + \mathbf{x}_1 + \mathbf{n}_2, \tag{3}$$

where $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{C}^n$ are the transmitted messages and $\mathbf{n}_1, \mathbf{n}_2 \sim \mathcal{CN}(0, \beta \mathbf{I}_n)$ is Gaussian noise. No fading is assumed in this scenario; i.e., $\mathbf{h}$ values are set to 1 for each link. The individual cross-entropy loss functions of the first and second transmitter-receiver pairs are $l_1 = -\log\left(\left[\hat{\mathbf{s}}_1\right]_{s_1}\right)$ and $l_2 = -\log\left(\left[\hat{\mathbf{s}}_2\right]_{s_2}\right)$ for the first and second autoencoder, respectively.

$\tilde{L}_1(\boldsymbol{\theta}_t)$, and $\tilde{L}_2(\boldsymbol{\theta}_t)$ correspond to the associated losses for mini-batch t. For joint training, dynamic weights α_t are adapted for each mini-batch t as

$$\alpha_{t+1} = \frac{\tilde{L}_1(\boldsymbol{\theta}_t)}{\tilde{L}_1(\boldsymbol{\theta}_t) + \tilde{L}_2(\boldsymbol{\theta}_t)}, \quad t > 0\,, \tag{4}$$

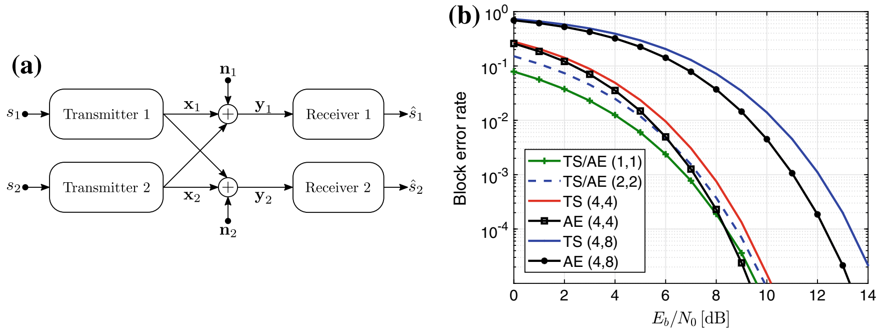


Fig. 9 **a** The two-user interference channel seen as a combination of two interfering autoencoders (AEs) that try to reconstruct their respective messages, **b** BLER versus E_b/N_0 for the two-user interference channel achieved by the autoencoder and $2^{2k/n}$-QAM time-sharing (TS) for different parameters (n, k)

where $\alpha_0 = 0.5$. Thus, the smaller $\tilde{L}_1(\boldsymbol{\theta}_t)$ is compared to $\tilde{L}_2(\boldsymbol{\theta}_t)$, the smaller is its weight α_{t+1} for the next mini-batch.

Figure 9b shows the BLER of one of the autoencoders as a function of E_b/N_0 for the sets of parameters $(n, k) = \{(1, 1), (2, 2), (4, 4), (4, 8)\}$. The DNN architecture for both autoencoders is the same as that provided in Table 1 by replacing n by $2n$. An average power constraint is used to be competitive with higher-order modulation schemes; i.e., allow varying amplitude in the constellation points for increasing data rate. As a baseline, uncoded $2^{2k/n}$-QAM (which has the same rate when used together with time-sharing between both transmitters) is considered. For (1, 1), (2, 2), and (4, 4), each transmitter sends a 4-QAM (i.e., QPSK) symbol on every other channel use. For (4, 8), 16-QAM is used instead. While the autoencoder and time-sharing have identical BLER for (1, 1) and (2, 2), the former achieves substantial gains of around 0.7 dB for (4, 4) and 1 dB for (4, 8) at a BLER of 10^{-3}.

The learned message representations at each receiver are shown in Fig. 10. For (1, 1), the transmitters have learned to use BPSK-like constellations (see Fig. 3a) in orthogonal directions (with an arbitrary rotation around the origin). This achieves the same performance as QPSK with time-sharing. However, for (2, 2), the learned constellations are not orthogonal anymore and can be interpreted as some form

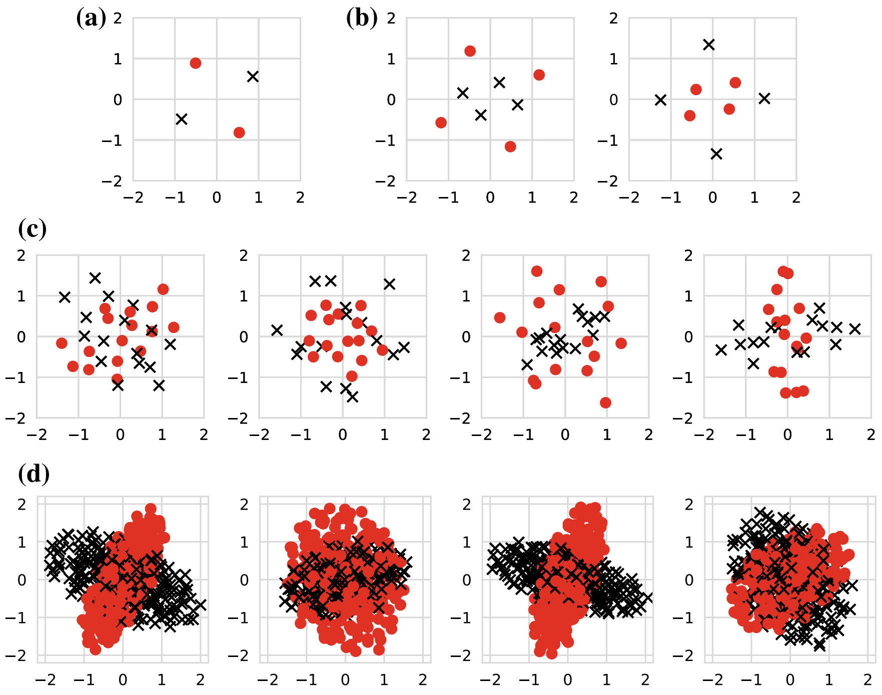


Fig. 10 Learned constellations for the two-user interference channel with parameters **a** (1, 1), **b** (2, 2), **c** (4, 4), and **d** (4, 8). The constellation points of Transmitters 1 and 2 are represented by red dots and black crosses, respectively [14]

of superposition coding. For the first symbol, Transmitter 1 uses high power and Transmitter 2 uses low power. For the second symbol, the roles are changed. For (4, 4) and (4, 8), the constellations are more difficult to interpret, but it can be seen that the constellations of both transmitters resemble ellipses with orthogonal major axes and varying focal distances. This effect is more visible for (4, 8) than for (4, 4) because of the increased number of constellation points.

Take-away: This section showed that deep learning-based autoenconder can be effectively used to develop transmitter (modulation and coding) and receiver (demodulation and decoding) functions jointly by combating channel impairments and optimizing end-to-end communication performance in terms of error rates. This approach applies to single, multiple antenna, and multiuser systems.

3 Deep Learning for Spectrum Situation Awareness

Cognitive radio has emerged as a programmable radio that aims to learn from wireless communication data and adapt to spectrum dynamics. For that purpose, cognitive radio senses its operational radio frequency (RF) environment and adjusts its operating parameters (e.g., frequency, power, and rate) dynamically and autonomously to modify system operation and improve its performance, such as maximizing throughput, mitigating interference, facilitating interoperability, or accessing spectrum as a secondary user [33].

Channel modeling is important while developing algorithms to enable cognitive capabilities and evaluating the performance of the communication systems. Most signal processing algorithms applied to wireless communications assume compact mathematically convenient channel models such as AWGN, Rayleigh, or Rician fading channel (or fixed delay/Doppler profiles consisting of Rayleigh fading taps). These existing channel models generally parameterize channel effects in a relatively rigid way which does not consider the exact statistics of deployment scenarios. Furthermore, practical systems often involve many hardware imperfections and nonlinearities that are not captured by these existing channel models [14]. Channel estimation is also an important task for a communication system to recover and equalize the received signal (reversing the channel effects). A known training sequence is often transmitted at the transmitter and the receiver typically uses methods such as maximum likelihood or MMSE channel estimation techniques, derived under compact mathematical channel models, to estimate the channel, e.g., MMSE estimator is applied in (2) for channel estimation in Sect. 2.2.

To support situational awareness, it is important for cognitive radios to quickly and accurately perform signal detection and classification tasks across a wide range of phenomena. One example is the DSA application where there are primary (legacy) and secondary (cognitive) users. Secondary users use the spectrum in an opportunistic manner by avoiding or limiting their destructive levels of interference to the primary users in a given frequency band. Therefore, secondary users need to detect and classify the signals received during spectrum sensing reliably to identify whether there is any primary user activity, other secondary users, or vacant spectrum opportunities. Conventional signal detection and classification algorithms aim to capture specific signal features (i.e., expert features) such as cyclostationary features and are typically developed to achieve performance goals such as detection against specific signal types and under specific channel model assumptions (e.g., AWGN). Therefore, these conventional algorithms often lack the ability to generalize to different signal types and channel conditions, while deep learning can capture and adapt its operation to raw and dynamic spectrum data of a wide variety of signal signatures and channel effects (that feature-based machine learning algorithms may struggle to capture).

Deep learning approaches have been used to address the challenges associated with both channel modeling and estimation as well as signal detection and classification tasks. In the following subsections we first describe how channel modeling and estimation can be performed using deep learning methods. Next, we describe the CNN architectures that are used for signal detection and modulation classification. Finally, we describe how to use GANs to augment training data in spectrum sensing applications.

3.1 Channel Modeling and Estimation

The performance of communication systems can often benefit from being optimized for specific scenarios which exhibit structured channel effects such as hardware responses, interference, distortion, multi-path and noise effects beyond simplified analytic models or distributions. Moreover, the channel autoencoder systems described in Sect. 2 requires the statistical model for the channel to be as close as possible to what the operational system will experience during training in order to achieve optimal performance (i.e., the phenomena during training should accurately match the phenomena during deployment). However, accurately capturing all these effects in a closed-form analytical model is a challenging (and often infeasible) task. As a result, the channel is often represented using simplified models without taking real-world complexities into account. Recently, model-free approaches where the channel response is learned from data are proposed for real-time channel modeling using deep learning techniques. In particular, stochastic channel response functions are approximated using GANs [34, 35], variational GANs [36], reinforcement learning and sampling approach [37], stochastic perturbation techniques [38], and reinforcement learning policy gradient methods [20].

GANs [39] have been successfully used for a number of applications such as generating fake images (e.g., faces, cats) to confuse image recognition systems. Recently, GANs have also been used in a wide range of applications such as audio generation, approximation of difficult distributions, and even the (human-guided) generation of novel art. Building upon this same idea, the GAN was applied to approximate the response of the channel in any arbitrary communication system in [34] and the resulting system was generally called a *Communications GAN*. The block diagram of the Communications GAN that learns a communication system over a physical channel with no closed-form model or expression is shown in Fig. 11.

As opposed to the original autoencoder shown in Fig. 4, a channel model with an analytic expression is not included in the autoencoder in Fig. 11. Two forms of the channel $h(\mathbf{x})$ are included instead to encompass modeling of any black-box channel transform where $\mathbf{x}$ is the transmitter output: $h_0(\mathbf{x})$ is a real-world physical measurement of the response of a communication system comprising a transmitter, a receiver, and a channel and $h_1(\mathbf{x}, \theta_\mathbf{h})$ is a non-linear DNN which seeks to mimic the channel response of h_0 synthetically, and is differentiable. $\theta_\mathbf{h}$ is the channel approximation of neural network parameters. During training, an iterative approach is used to reach an optimized solution, cycling between competing training objectives, updating weights for each network during the appropriate stage with manually tuned learning rates and relatively small networks for f, g, and h, and employing several fully connected ReLU layers for each. The physical channel $h_0(\mathbf{x})$ was implemented using an SDR (Universal Software Radio Peripheral, USRP B210 [40]), for over-the-air transmission tests. It was shown that an effective autoencoder-based communication system with robust performance can be learned by using an adversarial approach to approximate channel functions for arbitrary communications channel. This approach eliminates the need for a closed-form channel model reducing the need for assumptions on the form it takes.

The channel network $\mathbf{y} = h(\mathbf{x})$ is treated as a stochastic function approximation and the accuracy of the resulting conditional probability distribution $p(\mathbf{y}|\mathbf{x})$ is optimized in [36]. The channel approximation network $\hat{\mathbf{y}} = h(\mathbf{x}, \theta_\mathbf{h})$ is considered to be a conditional probability distribution, $p(\hat{\mathbf{y}}|\mathbf{x})$ and the distance between the conditional probability distributions $p(\mathbf{y}|\mathbf{x})$ and $p(\hat{\mathbf{y}}|\mathbf{x})$ resulting from the measurement

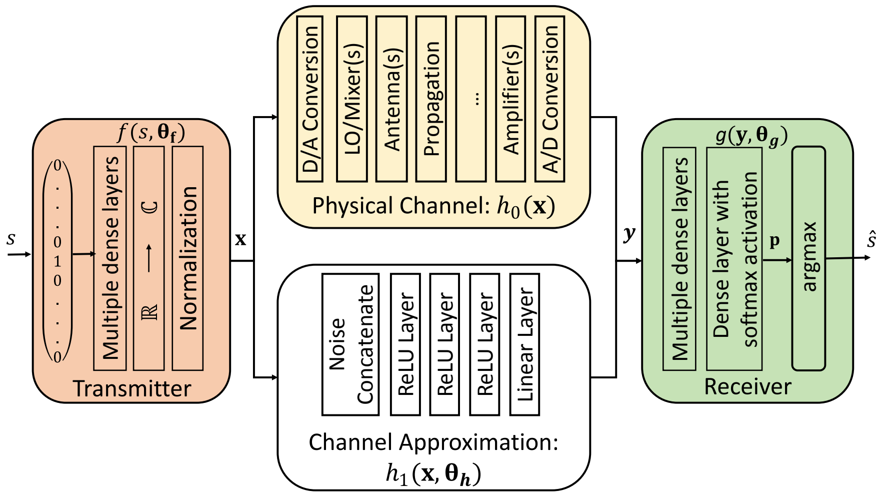


Fig. 11 A GAN for learning a communication system over a physical channel with no-closed form model

and from the variational channel approximation network are minimized. As in [39], the parameters of each network are minimized using the two stochastic gradients given in (5) and (6).

$$\nabla_{\theta_{\mathbf{D}}} \frac{1}{N} \sum_{i=0}^{N} \left[\log\left(D(x_i, y_i, \theta_{\mathbf{D}})\right) + \log\left(1 - D(x_i, h(x_i, \theta_{\mathbf{h}}), \theta_{\mathbf{D}})\right) \right], \tag{5}$$

$$\nabla_{\theta_{\mathbf{h}}} \frac{1}{N} \sum_{i=0}^{N} \log\left(1 - D(x_i, h(x_i, \theta_{\mathbf{h}}), \theta_{\mathbf{D}})\right). \tag{6}$$

A new discriminative network $D(x_i, y_i, \theta_{\mathbf{D}})$ is introduced to classify between real samples, $\mathbf{y}$, and synthetic samples, $\hat{\mathbf{y}}$, from the channel given its input, $\mathbf{x}$. $\theta_{\mathbf{D}}$ is the discriminative network parameters. $h(\mathbf{x}, \theta_{\mathbf{h}})$ takes the place of the generative network, $G(z)$, where $\mathbf{x}$ reflects conditional transmitted symbols/samples. N is the number of samples. Additional stochasticity in the function is introduced through variational layers. Furthermore, training such an arrangement using the improved Wasserstein GAN approach with gradient penalty (WGAN-GP) [41] allows convergence with minimal tuning.

Adam [19] optimizer is used with a learning rate between 10^{-4} and 5×10^{-4} to iteratively update the network parameters. The variational architecture for the stochastic channel approximation network is shown in Fig. 12a.

For performance evaluation, a communication system that transmits 1 bit/symbol is considered. A Chi-squared distributed channel model is assumed to explore a more uncommon channel scenario. The measured and approximated conditional distributions from the black box channel model are shown in Fig. 12b. There is some difference between the original distribution and its approximation, resulting partially from its representation as a mixture of Gaussian latent variables; however, this can

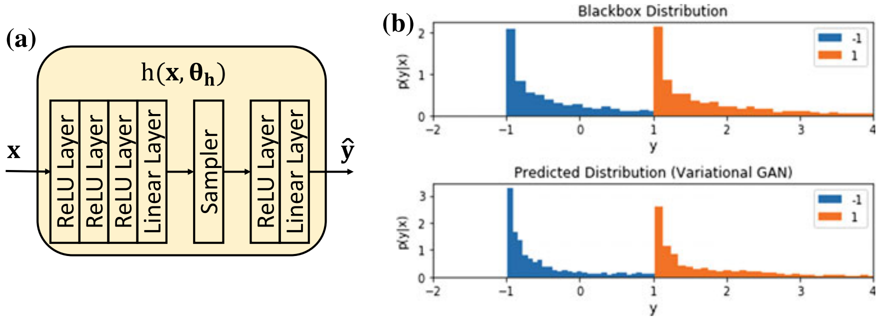


Fig. 12 **a** Variational architecture for the stochastic channel approximation network (conditional generator), **b** Learned one-dimensional distributions of conditional density on non-Gaussian (Chi-Squared) channel effects using variational GAN training [36]

be alleviated by choosing different sampling distributions and by increasing the dimensions of the latent space (at the cost of increased model complexity).

This approach can also capture more complex distributions such as the channel responses of cascades of stochastic effects by jointly approximating the aggregate distribution with the network. Consider a 16-QAM system that includes AWGN effects along with phase noise, phase offset, and non-linear AM/AM and AM/PM distortion effects introduced by a hardware amplifier model. Figure 13 illustrates the marginalized $p(\mathbf{x})$ distribution for both the measured version of the received signal, and the approximated version of the distribution when a stochastic channel approximation model is learned with variational GANs. It is observed that each constellation point's distribution, circumferential elongation of these distributions due to phase noise at higher amplitudes, and generally the first order approximation of the distribution are learned successfully.

On the receiver side, typically synchronization is performed on the signal (timing estimation, frequency offset estimation, etc.) before performing additional signal processing steps for conventional communication systems (e.g., symbol detection). Synchronization typically estimates these time, frequency, phase, and rate errors in the received data and corrects for them to create a normalized version of the signal. Learned communication systems described in Sect. 2 can in some instances perform implicit synchronization and channel estimation since hardware and channel impairments such as synchronization offsets can be included during training. From a learning perspective, we can treat these corrections as transforms, leveraging expert knowledge about the transforms to simplify the end-to-end task, but still allowing the estimators to be fully learned. This approach of radio transformer networks (RTNs), as explored in both of [14, 42], are shown to reduce training time and complexity and improve generalization by leveraging domain knowledge.

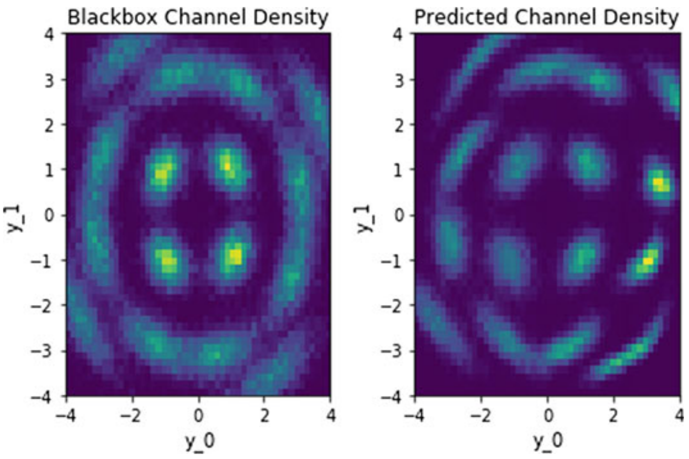


Fig. 13 Learned two-dimensional distributions of received 16-QAM constellation non-linear channel effects using variational GAN [36]

These offset effects exist in any real system containing transmitters and receivers whose oscillators and clocks are not locked together.

Timing and symbol-rate recovery processes involve the estimation and re-sampling of the input signal at correct timing offsets and sampling increments, which has a direct analogue to the extraction of visual pixels at the correct offset, shift or scale (e.g., applying the correct Affine transformation) in computer vision using transformer networks. The input data can be represented as a two-dimensional input, with the rows containing in-phase (I) and quadrature (Q) samples and N columns containing samples in time. A full 2D Affine transformation allows for translation, rotation, and scaling in 2D given by a 2×3 element parameter vector. To restrict this to 1D translation and scaling in the time dimension, the mask in (7) is introduced such that a normal 2D Affine transform implementation may be used from the image domain. θ_0, θ_1, and θ_2 are the remaining unmasked parameters for the 1D Affine transform:

$$\begin{bmatrix} \theta_0 & 0 & \theta_2 \\ 0 & \theta_1 & 0 \end{bmatrix} \tag{7}$$

Phase and frequency offset recovery tasks do not have an immediate analogue in the vision domain. However, a simple signal processing transform can be applied to accomplish these. The input signal is mixed with a complex sinusoid with phase and frequency as defined by two new unknown parameters as shown in (8).

$$y_n = x_n \, \mathrm{e}^{\,j(n\theta_3 + \theta_4)} \tag{8}$$

This transform can be directly implemented as a new layer in Keras [31], cascaded before the Affine transform module for timing and symbol-rate recovery.

The task of synchronization then becomes the task of parameter estimation of θ_i values passed into the transformer modules. Domain appropriate layers are used to assist in estimation of these parameters, namely, complex convolutional 1D layer and complex to power and phase layers. Although many architectures are possible, both the complex convolution operation and the differentiable Cartesian to Polar operation are used to simplify the learning task. Figure 14 shows one example of an

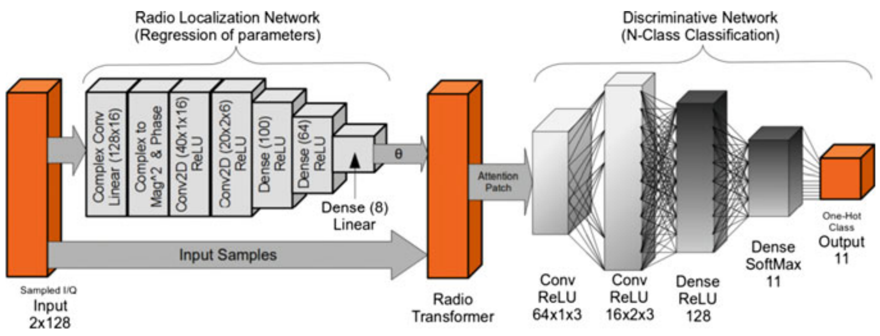


Fig. 14 RTN architecture [42]

RTN architecture. A dropout rate such as 0.5 can used between layers to prevent over-fitting, and Adam [19] SGD can be used to optimize network parameters on the training set, in this case with batch size 1024, and learning rate 0.001.

The density plots for pre- and post-transformed input constellations are shown in Fig. 15. When the constellation density for 50 test examples over a range of 20 time samples are observed, the density starts to form around the constellation points after using the radio attention model.

In both [14, 20], Rayleigh block fading channel is considered as the channel and RTNs are used for channel estimation. Then the received signal is divided by the learned channel response to equalize the input signal, which leads to improved SER performance, providing a more quantitative study of the RTN efficacy.

The described channel modeling approaches may be used broadly for enhanced optimization, test, and measurement of communication systems and specifically to provide effective model-free methods for various wireless tasks such as channel learning in autoencoder-based communications (see Sect. 2) and signal classification (see Sect. 3.2). Moreover, the developed RTN models can be used to extract the channel features, similar to channel estimation in conventional systems, and perform equalization by using a transformation layer which allows for imparting of expert knowledge without over-specifying learned models (e.g., writing estimators for specific protocols or references).

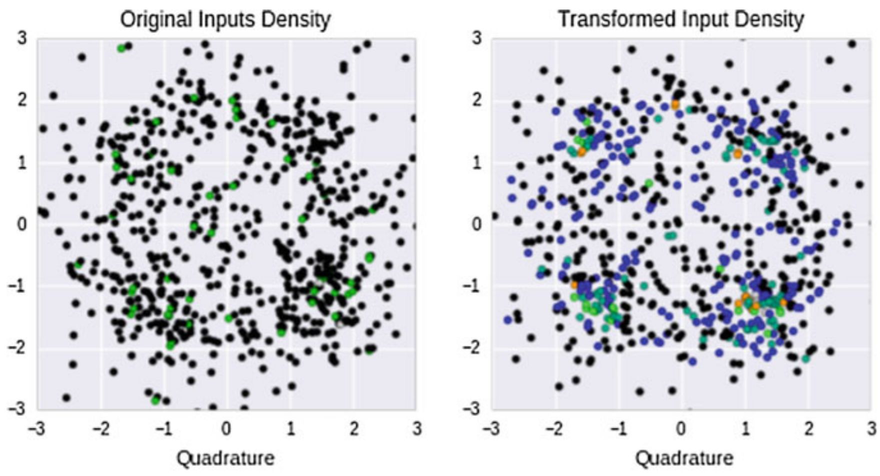


Fig. 15 Density plots of the pre- and post-transformed input constellations [42]

3.2 Signal Detection and Modulation Classification

Signal detection and classification functionalities define the ability of a wireless communication system to accurately build and maintain an up-to-date view of their current operating environment. Detecting and coexisting with other users of the spectrum, detecting and isolating sources of interference, flagging significant spectral events, or identifying spectral vacancies within the radio spectrum rely on signal detection and classification. The probability of detection is proportional to the SNR at the receiver. Traditionally, specific signal detectors are needed for each waveform, developed based on its analytic properties, resulting in systems which can be difficult to develop and deploy robustly in real-world wireless applications largely due to their over-specificity, complexity, or sub-optimal performance in real world conditions [43].

The RF spectrum is shared with many different signal types ranging from TV broadcast to radar. Signal detection and classification tasks are particularly challenging in the presence of multiple waveforms operating at the same frequency and at low SNR. Conventional signal detection and classification methods can be categorized as:

– *General methods*: These methods do not require any prior information on the signal types. They detect multiple signal types; however, their constant false alarm rate (CFAR) performance is relatively poor. Energy detector [44] is an example of detectors which do not require prior information. These type of detectors can be easily cast into convenient probabilistic form for analysis, but they are severely constrained in their abilities to leverage additional information about signal context or structure to improve performance.
– *Specialized methods*: These methods provide sensitive detectors for specific signal types. The detection and classification methods are developed using specific features of the signal of interest. Matched filters and cyclostationary signal detectors [44] are examples to this type. These methods are often not scalable since a new type of classifier is required for each new waveform.

A new class of deep learning-based radio waveform detectors that leverages the powerful new techniques developed in computer vision, especially convolutional feature learning, holds the potential to improve the signal detection and classification performance of practical systems by generalizing well and remaining sensitive to very low power signals [43]. A strong analogy of this task exists in computer vision with object identification and localization tasks. Recent object detection and localization approaches associate specific object classes with bounding box labels within the image. A similar approach was followed in [45], where the RF spectrum is represented as an image and CNNs are used to detect, localize and identify radio transmissions within wide-band time-frequency power spectrograms using feature learning on 2D images.

Gradient-weighted Class Activation Mapping (Grad-CAM) uses the gradients of any target concept flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image that aids to predict the concept [46]. Grad-CAM is used to perform the spectral event localization in [45]. Figure 16 shows the block-diagram of the Grad-CAM, which is used for spectral event localization. The gradient of activation score y^C (instead of the class probability) is calculated with respect to all the feature maps of a given convolution layer based on the provided input label C. The global average pooling [47] of the gradients gives the corresponding weight associated with the feature map. Finally, the weighted sum of the feature maps is passed through an element-wise ReLU unit to get the class activation map.

To demonstrate the performance in this work, a dataset was collected in 13 different frequency bands using a USRP B205 transceiver at eight different locations across five distinct cities and across a range of different bands and traffic patterns. Signal types in the dataset include GSM, LTE, ISM, TV, and FM among others. Spectrogram plots shown in Fig. 17, labeled as *input spectrum*, are generated using the collected data to show the signal strength over time and frequency. The x-axis shows the time and the y-axis shows the signal frequency. These images are used as an input to the CNN architecture. The Grad-CAM implementation results are also shown in Fig. 17. A hot region of activation is observed on top of the signal bursts, as expected. The trained feature objective was to classify the band instead of activating all instances of a certain emission type since the labels for every signal activity in a band are not provided; i.e., each spectrogram is assigned only one label even though there may be some other narrow band signals in the same spectrogram. For this reason, for some examples, the activation map highlights only strong parts of the signal and some parts of the signals are favored for identification.

Figure 18a shows the confusion matrix for the classification results. This method for detecting, classifying and localizing emissions within a spectrogram provides reasonable classification performance and reasonable class activation maps corresponding to activity regions in most cases as pictured.

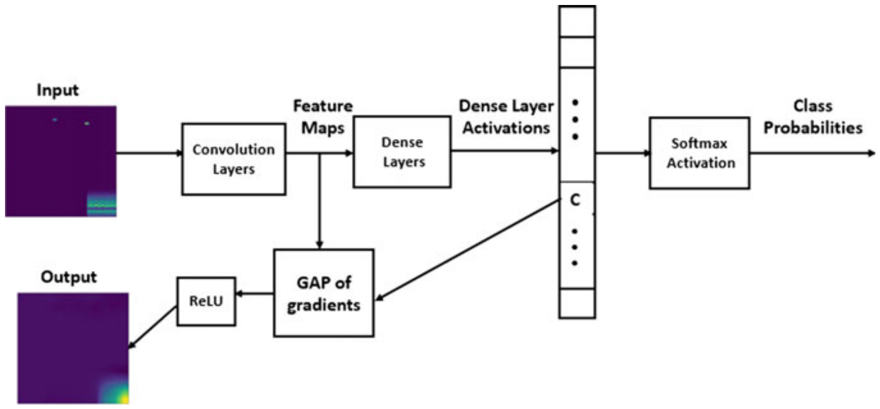


Fig. 16 Block diagram of Grad-CAM [46]

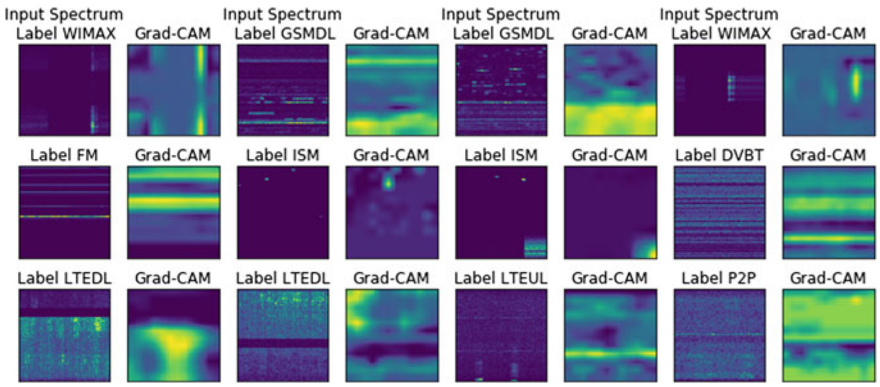


Fig. 17 GradCAM based activation maps and corresponding input spectrograms for 12 test examples from the dataset [45]

For the task of supervised modulation recognition, a number of other non-NN based machine learning techniques from literature were compared with that of a convolutional deep learning architecture in terms of performance. In [3], the generated data set consists of 11 modulations: 8 digital and 3 analog modulations, which are all widely used in wireless communication systems. These consist of BPSK, QPSK, 8PSK, 16-QAM, 64-QAM, BFSK, CPFSK, and PAM4 as digital modulations, and WB-FM, AM-SSB, and AM-DSB as analog modulations. Data is modulated at a rate of roughly 8 samples per symbol with a normalized average transmit power of 0 dB. These signals are exposed to realistic channel effects. Thermal noise results in relatively flat white Gaussian noise at the receiver which forms a noise floor or sensitivity level and SNR. Oscillator drift due to temperature and other semiconductor physics differing at the transmitter and receiver result in symbol timing offset, sample rate offset, carrier frequency offset, and phase difference. These effects lead to a temporal shifting, scaling, linear mixing/rotating between channels, and spinning of the received signal based on unknown time varying processes. Moreover, real channels undergo random filtering based on the arriving modes of the transmitted signal at the receiver with varying amplitude, phase, Doppler, and delay. This is a phenomenon commonly known as multi-path fading or frequency selective fading, which occurs in any environment where signals may reflect off buildings, vehicles, or any form of reflector in the environment.

Figure 18b shows a simple CNN architecture used for the modulation classification task, an un-tuned 4-layer network utilizing two convolutional layers and two (overly sized) dense fully connected layers. Layers use ReLU activation functions except for a softmax activation on the output layer to act as a classifier. Dropout regularization is used to prevent over-fitting, while a $\|W\|_2$ norm regularization on weights and $\|\mathbf{h}\|_1$ norm penalty on dense layer activations can also encourage sparsity of solutions [48, 49]. Training is conducted using a categorical cross-entropy loss and an Adam [19] solver.

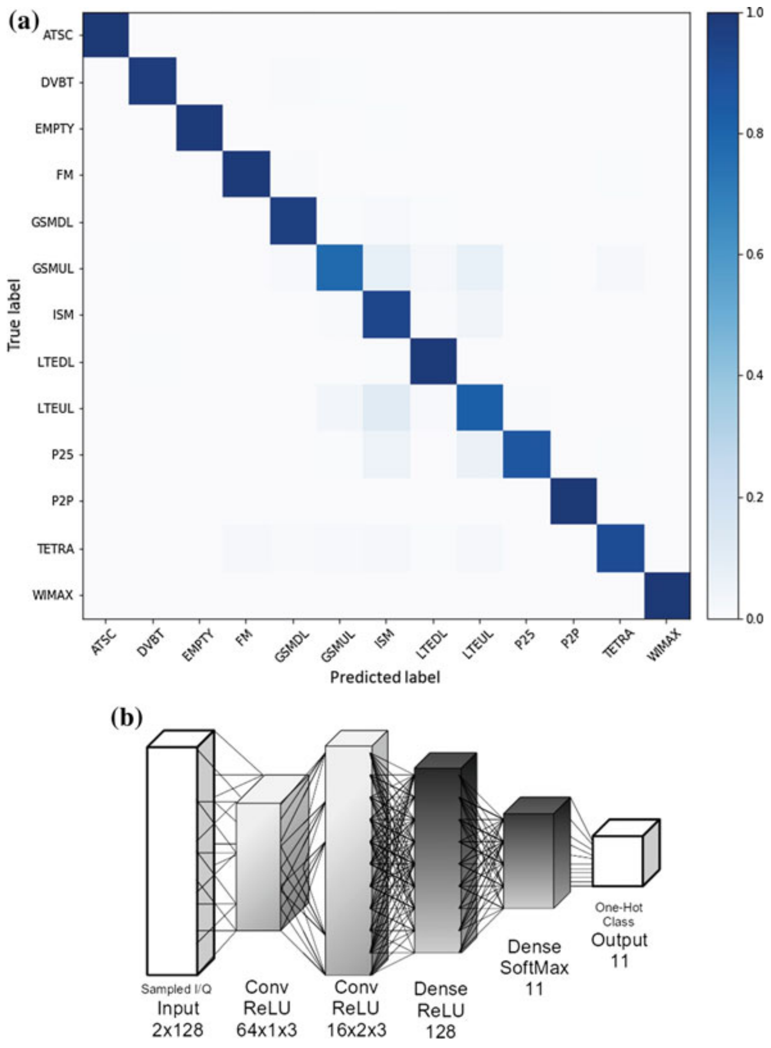


Fig. 18 **a** Confusion matrix for RF band classification [45], **b** CNN architecture [3]

Expert features (Higher order moments, and cumulants) are used by the baseline classifiers. Figure 19 shows the performance results of the Naive Bayes, support vector machine (SVM) and CNN network architecture results where the CNN classifier outperforms the Naive Bayes and SVM classifiers at all SNRs.

For more realistic evaluations, over-the-air dataset was generated in [50] and the modulation classification performance was compared between *virtual geometry group* (VGG) and *residual networks* (RNs) with better architecture tuning, as well as a stronger XGBoost based baseline. It was shown that the RN approach achieves state-of-the-art modulation classification performance on for both synthetic and over-

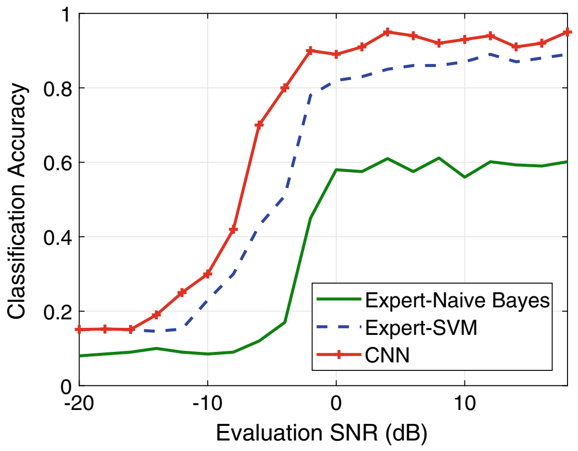


Fig. 19 Example of deep learning outperforming conventional machine learning in wireless domain. CNN is more successful than SVM and Naive Bayes in classifying a variety of digital modulations (BPSK, QPSK, 8PSK, 16-QAM, 64-QAM, BFSK, CPFSK, and PAM4) and analog modulations (WB-FM, AM-SSB, and AM-DSB) [3]

the-air signals using datasets consisting of 1 million examples, each 1024 samples long. The RN achieves roughly 5 dB higher sensitivity for equivalent classification accuracy than the XGBoost baseline at low SNRs while performances are identical at low SNRs. At high SNRs, a maximum classification accuracy rate of 99.8% is achieved by the RN, while the VGG network achieves 98.3% and the baseline method achieves a 94.6% accuracy.

3.3 Generative Adversarial Methods for Situation Awareness

Radios collect spectrum data samples such as raw (complex-valued) data samples or received signal strength indicator (RSSI) values through spectrum sensing, and use them to train DNNs for various applications such as channel estimation or waveform classification, as discussed in previous sections. There are two important hurdles to overcome before using spectrum data for deep learning purposes.

1. Deep learning requires a large number of data samples to be able to train the complex structures of DNNs. This may not be readily available via spectrum sensing, since a wireless user who spends too much time on spectrum sensing may not have enough time left for other tasks such as transmitting its data packets. Therefore, there may not be enough number of wireless data samples available to train a DNN. *Training data augmentation* is needed to expand the training data collected in spectrum sensing.
2. Characteristics of spectrum data change over time as the underlying channels, interference and traffic effects, as well as transmit patterns of wireless users change. Therefore, training data collected for one instant may not be fully applicable in another instant. One example is the channel change when the wireless nodes move from outdoors to indoors, where more multipaths and therefore dif-

ferent channel conditions are expected. *Domain adaptation* is needed to change test or training data collected in spectrum sensing from one domain (e.g., low mobility) to another domain (high mobility).

The GAN has emerged as a viable approach to generate synthetic data samples based on a small number of real data samples in a short learning period and augment the training data with these synthetic data samples for computer vision, text, and cyber applications [51–53]. The GAN consists of a generator and a discriminator playing a minimax game. The generator aims to generate realistic data (with labels), while the discriminator aims to distinguish data generated by the generator as real or synthetic. Conditional GAN extends the GAN concept such that the generator can generate synthetic data samples with labels [54]. Figure 20 shows the conditional GAN architecture. When applied to wireless communications, the GAN needs to capture external effects of channel patterns, interference, and traffic profiles in addition to waveform features. The GAN has been applied for training data augmentation for channel measurements in spectrum sensing [55], modulation classification [56], jamming [57, 58], and call data records for 5G networks [59].

As an example, consider an adversary that senses the spectrum and observes transmissions of another node (hidden in channel impairments, traffic on/off patterns and other background transmissions). Based on these observations, the adversary trains a DNN to predict when there will be a successful transmission and jams it. See Sect. 4 for details of this setting when deep learning for wireless communications security is discussed. If the adversary waits too long to collect data, it may lose the opportunity to jam transmissions. Therefore, the adversary collects a small number of sensing samples and then augments them through GAN.

The wireless application of GAN for domain adaptation has remained limited so far. Reference [55] studied the adaptation of training data for spectrum sensing, where a wireless receiver decides if there is an active transmitter (label 1) or not (label 2). There are two environments corresponding to two different channel types, namely Rayleigh fading distributions with variance 0.2 (environment 1) and 2 (environment 2). Assume the receiver has training data for environment 1 and trained a classifier, whereas there is no training data for environment 2. Therefore, the receiver generates synthetic training data samples for environment 2. Training data adaptation consists of a bidirectional GAN [60], a conditional GAN [54], and a classifier. Bidirectional GAN obtains the inverse mapping from data to the conditioned noise by using a GAN and an autoencoder that together learn to take the inverse of a neural network. As the environment changes from 1 to 2, a new conditional GAN is trained that

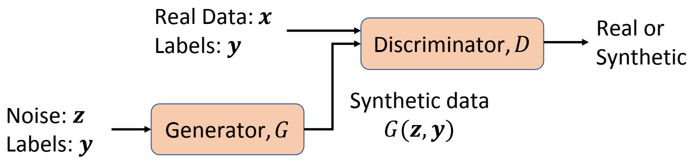


Fig. 20 Conditional GAN for training data augmentation

takes the new samples in environment 2 as real inputs. Instead of random noise as synthetic inputs, the inverse mapping of the bidirectional GAN is used and the labels in environment 1 is carried to environment 2 to train the CGAN. After CGAN training, a classifier is trained with domain adapted samples and used to label new samples collected in environment 2. This approach prevents 42% drop in accuracy of SVM-based spectrum sensor operating at 5 dB SNR [55].

Separately, the GAN was used in [61] to match waveform, channel, and radio characteristics, and spoof wireless signals that cannot be reliably distinguished from legitimate signals. This attack can be used against signal authentication systems and can be launched to emulate primary user behavior in primary user emulation (PUE) attacks.

Take-away: This section showed that deep learning provides novel means to characterize and analyze the spectrum. By outperforming conventional machine learning algorithms, DNNs significantly contribute to spectrum situation awareness for channel modeling and estimation with GANs and FNNs and signal detection and classification with CNNs.

4 Deep Learning for Wireless Communications Security

Wireless communications are highly susceptible to security threats due to the shared medium of wireless transmissions. A typical example of wireless attacks is the *jamming* attack that aims to disrupt wireless communications by imposing interference at receivers (e.g., see [62]) and causing denial of service (DoS) [63]. These attacks use different communication means (e.g., power control [64] or random access [65]) and apply at different levels of prior information on attacker's intent [66]. As radios become smarter by performing more sophisticated tasks, they also become vulnerable to advanced attacks that target their underlying tasks. One example is the *spectrum sensing data falsification* (SSDF) attack, where an adversary that participates in cooperative spectrum sensing deliberately falsifies its spectrum sensing result (namely, whether the channel is busy or idle) [67]. This way, the adversary aims to change the channel occupancy decision from busy to idle (such that the subsequent transmission fails) or from idle to busy (such that no other radio transmits and either the transmission opportunity is wasted or the adversary gets the opportunity to transmit). Data falsification may also occur at other network functions. One example is that routing decisions are manipulated by falsifying measures of traffic congestion (such as queue backlogs) exchanged throughout the wireless network [68, 69].

Beyond these security threats, the increasing use of deep learning by radios opens up opportunities for an adversary to launch new types of attacks on wireless communications. In particular, deep learning itself becomes the primary target of the adversary. The paradigm of learning in the presence of an adversary is the subject of the emerging field of *adversarial machine learning* [70] that has been traditionally applied to other data domains such as computer vision. The *exploratory (inference) attack* [71] is one example, where the adversary tries to learn the inner-workings of a machine learning classifier (such as a DNN) by querying it with some data samples, collecting the returned labels, and building a functionally equivalent classifier.

Adversarial machine learning provides the necessary optimization mechanisms to launch and mitigate attacks on machine learning. In addition to exploratory attacks, two other popular types of attacks are evasion and causative (poisoning) attacks. In *evasion attacks*, the adversary selects or generates data samples to query a machine learning algorithm such as a deep learning classifier and fool it into making wrong decisions [72]. In *causative attacks*, the adversary targets the training process and tampers with the training data (i.e., modifies the corresponding labels) such that the machine learning algorithm is not trained adequately [73]. As deep learning is sensitive to errors in training data, this attack is effective against DNNs. While these attacks have been successfully applied in different data domains such as computer vision (such as image classification [53]) and natural language processing (such as document classification [74]), they cannot be readily applied in wireless communications. The reasons are multi-fold:

– The adversary does not have a mechanism to directly query a wireless transmitter but it can only observe its transmission characteristics over the air.
– The collection of training data by the adversary is through a noisy channel, i.e., the training data of the adversary is imperfect by default.
– The training data and labels of the adversary and its target are different in wireless domain. Their data samples are different because they are received through different channels, whereas their labels are different because their machine learning objectives are different. For example, a transmitter may try to detect whether the channel is busy, while the jammer may try to predict when there will be a successful transmission.

Hence, the application of adversarial machine learning to wireless domain is not trivial and needs to account for the aforementioned differences, both from the attacker and defender perspectives [57, 58, 75]. As shown in Fig. 21, a basic communication scenario is used to illustrate wireless attacks based on adversarial machine learning [57]. There is one cognitive transmitter T that acts as a secondary user and dynamically accesses the spectrum to communicate with its receiver R while avoiding interference from a background transmitter B that acts as a primary user (e.g., TV broadcast network). T uses a decision function such as a deep learning classifier for its transmissions to capture B's transmission pattern as well as channel effects. There is also an adversary A that does not know the decision function of T and tries to learn it by sensing the spectrum. This corresponds to a *black-box* exploratory attack that is followed by other attacks such as jamming to reduce the performance of T. In the following, we will describe the exploratory attack on wireless communications and how it is used to launch an effective jamming attack [57]. Then we will present other wireless attacks motivated by adversarial deep learning and discuss defense strategies.

4.1 Operational Modes for Transmitter and Adversary

A synchronized slotted time is assumed where all nodes operate on a single channel (with fixed center frequency and instantaneous bandwidth). Channel gain between any transmitting node i (T, B, or A) and any receiving node j (R, T, or A) is given by $h_{ij}(t)$ in time slot t. Then, j receives signal

$$y_j(t) = \sum_{i \in \mathcal{T}(t)} h_{ij}(t) x_i(t) + n_j(t) \tag{9}$$

in time slot t, where $\mathcal{T}(t)$ is the set of transmitting nodes, $n_j(t)$ is the receiver noise at j, and $x_i(t)$ carries a signal if $i \in \mathcal{T}(t)$, otherwise $x_i(t) = 0$. Since channel and noise realizations at A (namely, $h_{BA}(t)$ and $n_A(t)$) and T (namely, $h_{BA}(t)$ and $n_A(t)$) are different, they observe different data input for their tasks. It is assumed that $n_j(t)$ is random according to a zero-mean Gaussian distribution with power normalized as one, and $h_{ij}(t)$ depends on the distance d_{ij} between i and j and type of fading. It is also assumed that signal strength diminishes proportionally to $1/d_{ij}^2$ and log-normal shadowing is used as the shadowing model (namely, flat fading is considered such that the coherence bandwidth of the channel is larger than the bandwidth of the signal and all frequency components of the signal experience the same magnitude of fading). Note that $y_j(t)$ is the signal received during data transmission or sensing periods. In the latter case, $y_j(t)$ is denoted as $s_j(t)$. Next, the operation modes of background transmitter B, transmitter T, receiver R, and adversary A are discussed, as illustrated in Fig. 21.

Background transmitter *B* The transmit behavior (idle or busy) of B determines the channel status (idle or busy) in each time slot. There are random packet arrivals at B according to the Bernoulli process with rate λ (packet/slot). If B is in idle status and has a packet to transmit, it is activated with certain probability and keeps transmitting until there is no packet anymore in its queue. Since B's busy/idle states are correlated

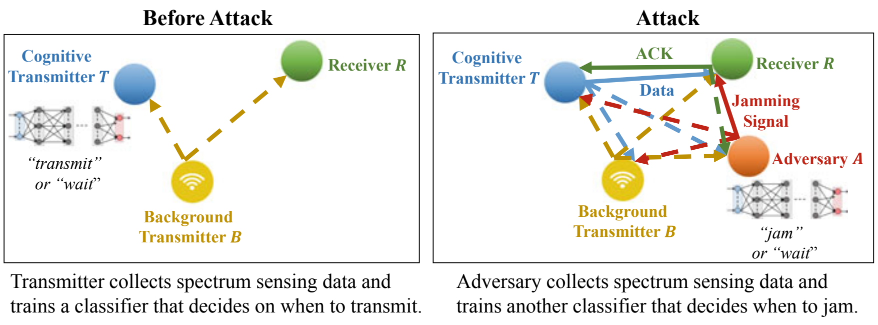


Fig. 21 Adversarial deep learning to launch a wireless attack

over time, both T and J need to observe not only the last channel status but the past channel states over several time slots to predict the current channel status.

Transmitter T In each time slot, T senses the channel and detects whether the channel status is idle or busy, i.e., whether B remains idle or transmits. If idle, T transmits data (a packet) to R in this time slot. T has trained a DNN (unknown to J) as the classifier C_T that classifies the current time slot t as idle or busy based on recent K_T sensing results $(s_T(t - K_T + 1), \ldots, s_T(t-1), s_T(t))$. In time slot t, the data sample for C_T is

$$\boldsymbol{s}_T(t) = (s_T(t - K_T + 1), \ldots, s_T(t-1), s_T(t)) \tag{10}$$

and the corresponding label is

$$L_T(t) = \{\text{“idle”}, \text{“busy”}\}, \tag{11}$$

where “idle” or “busy” means that the channel is idle or busy, respectively. Thus, the training data for C_T is built as $\{(\boldsymbol{s}_T(t), L_T(t))\}_t$. T obtains the label $L_T(t)$ of a sample only indirectly by observing whether its transmission (if any) is successful or not. A successful transmission indicates an idle channel and a failure indicates a busy channel. Note that this is a noisy observation since a transmission of T may fail or succeed depending on channel conditions even when B does not transmit or transmits, respectively. T deems a transmission as successful if it receives an acknowledgment (ACK) from R. If there is no ACK received, then T deems the transmission as failed. Note that T uses multiple sensing results as its features since features should be able to capture time correlation and help achieve a high sensing accuracy in a short period of time. Then classifier $C_T : \boldsymbol{s}_T(t) \mapsto L_T(t)$ defines the mapping from sensing results to occupancy decision and consequently to transmission decision in time slot t.

Adversary A Due to the open nature of wireless spectrum, A can also sense the spectrum and then predict whether there will be a successful transmission (with feedback ACK), or not (without a feedback) in a time slot. In the former case, A transmits to jam the channel in this time slot. In the latter case, A remains idle. Without knowing C_T, A builds another classifier C_A itself, which predicts whether there will be a successful transmission, or not, in time slot t based on recent K_A sensing results $(s_A(t - K_A + 1), \ldots, s_A(t-1), s_A(t))$. The goal of A is to infer C_T by building a surrogate classifier C_A. Note that A needs to learn relative channel effects and T's transmit behavior that in turn depends on B's transmit behavior and corresponding channel effects. This is a difficult learning task that needs to be handled in a black-box manner without any prior knowledge. Therefore, it is imperative for A to use a DNN as C_A. In time slot t, the data sample for C_A is

$$\boldsymbol{s}_A(t) = (s_A(t - K_A + 1), \ldots, s_A(t-1), s_A(t)) \tag{12}$$

and the corresponding label is

$$L_A(t) = \{\text{“ACK”}, \text{“no ACK”}\}, \tag{13}$$

where “ACK” or “no ACK” means that there is an ACK following a transmission, or not respectively. Thus, the training data for C_A is built as $\{(\boldsymbol{s}_A(t), L_A(t))\}_t$. C_A is defined as the mapping from sensing results to prediction of successful transmission and consequently to jamming decision in each time slot. A does not jam all time slots, although doing so can maximize the success of jamming, since A will be easily detected if it is jamming in all time slots due to the high false alarm rate and J may have power budget in terms of the average jamming power (thus it cannot jam all time slots).

Receiver *R* R receives a transmission of T successfully if the signal-to-interference-and-noise-ratio (SINR) is larger than some threshold β. SINR captures transmit power, channel, and interference effects. Whenever a transmission is successfully received, R sends an ACK back to T over the short ending period of the time slot. In the meantime, A senses the spectrum and potentially detects the presence of ACK (without decoding it) by considering the fact that ACK messages are typically distinct from data messages (they are short and they follow the data transmission with some fixed time lag).

4.2 Jamming Based on Exploratory Attack

Deep Learning by Transmitter *T* 1000 samples are collected by T and split by half to build its training and test data. 10 most recent sensing results are used to build one data sample (i.e., $K_T = 10$). T trains an FNN as C_T. The Microsoft Cognitive Toolkit (CNTK) [76] is used to train the FNN. T optimizes the hyperparameters of the DNN to minimize $e_T = \max\{e_T^{MD}, e_T^{FA}\}$, where e_T^{MD} is the error probability for misdetection (a time slot is idle, but T predicts it as busy) and e_T^{FA} is the error probability for false alarm (a time slot is busy, but T predicts it as idle). When the arrival rate λ for B is 0.2 (packet/slot), the optimized hyperparameters of C_T are found as follows. The neural network consists of one hidden layer with 100 neurons. The cross-entropy loss function is minimized to train the neural network with backpropagation algorithm. The output layer uses softmax activation. The hidden layers are activated using the sigmoid function. All weights and biases are initialized to random values in $[-1.0, 1.0]$. The input values are unit normalized in the first training pass. The minibatch size is 25. The momentum coefficient to update the gradient is 0.9. The number of epochs per time slot is 10.

In test time, C_T is run over 500 time slots to evaluate its performance. The positions of the T, R and B are fixed at locations (0, 0), (10, 0), and (0, 10), respectively. All transmit powers are set 30 dB above noise power. The SINR threshold β is set as 3. For these scenario parameters, $e_T^{MD} = e_T^{FA} = 0$. T makes 400 transmissions and 383 of them are successful. Note that 17 transmissions on idle channels fail due to random channel conditions. Thus, the throughput is $383/500 = 0.766$ packet/slot and the success ratio is $383/400 = 95.75\%$. Next, we will show how adversarial deep learning-based jammer can significantly reduce this performance.

Adversarial Deep Learning by Adversary A *Exploratory attack* aims to infer a machine learning (including deep learning) classifier and has been applied to other data domains such as text classification in [71] and to image classification in [72]. In these previous works, the adversary queries the target classifier, obtains labels of a number of samples and then trains a functionally equivalent classifier using deep learning. Two classifiers are functionally equivalent if they provide the same labels for the same sample. However, this approach cannot be applied to the wireless setting due to the differences in data samples and labels.

– Data samples at a given time are different, as T and A receive signals through different channels (i.e., due to different distances from B and realizations), such that spectrum sensing results $s_T(t)$ and $s_A(t)$ are different at any time t. At a given time t, the signal from B is received at T, R, and A as $y_T(t) = h_{BT}x_B(t) + n_T(t)$, $y_R(t) = h_{BR}x_B(t) + n_R(t)$, and $y_A(t) = h_{BA}x_B(t) + n_A(t)$, respectively where h_{BT}, h_{BR}, and h_{BA} are the channel gains and $n_T(t)$, $n_R(t)$, and $n_A(t)$ are the receiver noises.
– Classifiers of T and A have different types of labels. T's labels indicate whether the channel is busy or idle, whereas A's labels indicate whether T will have a successful transmission, or not.

A trains an FNN as the deep learning classifier C_A. For that purpose, 1000 samples are collected by A and split by half to build its training and test data. J uses the most recent 10 sensing results to build one data sample (i.e., $K_A = 10$). J aims to jam successful transmissions (with received ACK feedback) only. A optimizes the hyperparameters to minimize $e_A = \max\{e_A^{MD}, e_A^{FA}\}$, where e_A^{MD} is the error probability for misdetection (T's transmission is successful, but A predicts there will not be an ACK) and e_A^{FA} is the error probability for false alarm (T does not transmit or T's transmission fails (even without jamming), but A predicts that there will be an ACK). The training time (including hyperparameter optimization) is 67 s and the test time per sample is 0.024 ms. The optimized hyperparameters of the C_A are found as follows. The neural network consists of two hidden layers with 50 neurons. The cross-entropy loss function is used to train the DNN with backpropagation algorithm. The output layer uses softmax activation. The hidden layers are activated using the hyperbolic tangent (Tanh) function. All weights and biases are initialized to random values in $[-1.0, 1.0]$. The input values are unit normalized in the first training pass. The minibatch size is 25. The momentum coefficient to update the gradient is 0.9. The number of epochs per time slot is 10. With these hyperparameters, the error e_A is minimized to 1.48%. Note that the hyperparameter optimization affects the accuracy. For instance, if the number of layers is decreased to 1, the error e_A increases to 1.73%. Similarly, if the number of neurons per layer is changed to 30, the error e_A increases to 2.22%.

In test time, C_A is run over 500 time slots to evaluate its performance. The position of A is fixed at location (10, 10) and its jamming power is 30 dB above noise power.

If there is no jamming, T will have 383 successful transmissions. Under A's attack, the number of misdetections is 16, i.e., misdetection probability is $e_A^{MD} = 16/383 = 4.18\%$ (majority of successful transmissions are jammed), and the number of false alarms is 17, i.e., false alarm probability is $e_A^{FA} = 17/(500 - 383) = 14.53\%$. As the significant impact of this attack, there are only 25 successful transmissions among 400 transmissions. Thus, the throughput of T is reduced from 0.766 packet/slot to $25/500 = 0.05$ packet/slot and the success ratio of T is reduced from 95.75% to $25/400 = 6.25\%$.

As a benchmark, a conventional attack without adversarial deep learning is also considered. In this *sensing-based jamming*, A jams the channel if its received power during spectrum sensing in the current slot is greater than a threshold τ. Note that the performance of a sensing-based jammer relies on proper selection of τ. If τ is too low, the number of false alarms increases. If τ is too high, then the number of misdetections increases. Note that τ is usually given as a fixed value since there is no clear mechanism to select τ. For a performance upper bound, τ is selected as 3.4 that minimizes e_A and used to compute the throughput and the success ratio of the transmitter in the presence of sensing-based jammer. Then $e_A^{MD} = 12.8\%$ and $e_A^{FA} = 12.6\%$. Note that e_A^{MD} grows quickly to 30.0% when τ is increased to 5, whereas e_A^{FA} grows to 14.0% when τ is reduced to 2. With the best selection of τ, the throughput of T is reduced to 0.140 packet/slot and the success ratio of T is reduced from 16.99%. On the other hand, if τ is selected arbitrarily (say, 4.7), the throughput of T becomes 0.576 packet/slot and the success ratio of T becomes 69.90% (i.e., the attack is not as effective). The results which are summarized in Table 3 show the importance of adversarial deep learning in launching wireless jamming attacks.

Generative Adversarial Learning for Wireless Attacks In the training process of adversarial deep learning, A collected 500 samples to build its classifier C_A. From a practical attack point of view, it is critical to shorten this initial learning period of A before jamming starts. For that purpose, J builds the GAN to generate synthetic data samples based on a small number of real data samples in a short learning period. Then it uses these synthetic data samples to augment its training data, as discussed in Sect. 3.3.

The conditional GAN is implemented in TensorFlow [32] by using the FNNs with three hidden layers each with 128 neurons for both generator and discriminator of the GAN. Leaky ReLu is used as the activation function. Adam optimizer [19] is used as the optimizer to update the weights and biases. The output of each hidden layer

Table 3 Effect of different attack types on the transmitter's performance [57]

Attack type	Throughput	Success ratio (%)
No attack	0.766	95.75
Adversarial deep learning	0.050	6.25
Sensing-based attack ($\tau = 3.4$)	0.140	16.99
Sensing-based attack ($\tau = 4.7$)	0.576	69.90

is normalized (via batch normalization). Figure 22 shows the losses of generator and discriminator. Note that the losses fluctuate significantly when the GAN training starts and eventually converges after 3000 iterations of the GAN training process.

The similarity between real and synthetic data distributions are measured by the Kullback-Leibler (KL) divergence. The KL divergence is given by

$$D_{\mathrm{KL}}(P\|Q) = -\sum_{x\in\mathcal{X}} P(x)\log\left(\frac{Q(x)}{P(x)}\right) \tag{14}$$

for two distributions P and Q with the support over $\mathcal{X}$. Denote P as the distribution of synthetic data samples (generated by the GAN), Q as the distribution of real samples, and c as the random variable for the channel status ($c = 0$ if idle and $c = 1$ if busy). Define $P_i(x) = P(x|c = i)$ for $i = 0, 1$. Then, $D_{\mathrm{KL}}(P_0\|Q_0) = 0.1117$ and $D_{\mathrm{KL}}(P_1\|Q_1) = 0.1109$. The test time per sample is measured as 0.024 ms (much smaller than the channel coherence time). If sensing results are obtained per second and 500 measurements are made, it takes 500 s to collect 500 RSSI levels without using the GAN. It takes 23 s to train the GAN using a GeForce GTX 1080 GPU and generate 500 synthetic samples from the GAN. Since 10 real samples are collected over 10 s, it takes 33 s to prepare data with the GAN. Hence, the GAN significantly reduces the data collection time before A starts jamming. When A builds its classifier C_A based on 10 real data samples, the error probabilities are 19.80% for false alarm and 21.41% for misdetection. After adding 500 synthetic data samples, the error probabilities drop to 7.62% for false alarm and to 10.71% for misdetection, namely close to the levels when 500 real data samples are used to train the DNN.

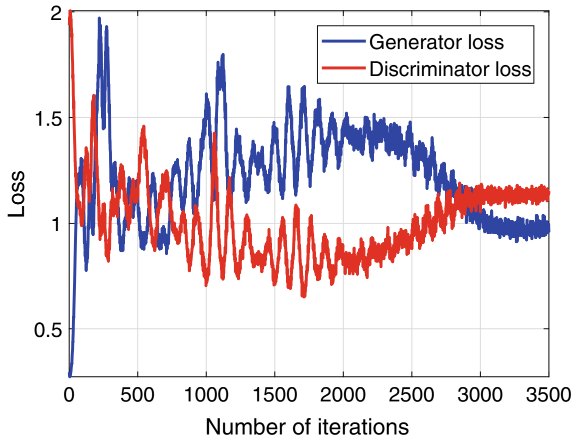


Fig. 22 Discriminator and generator losses during training [57]

4.3 Other Attacks Based on Adversarial Deep Learning

There are various other wireless attacks that can be launched through adversarial machine learning. A brief taxonomy of attacks from the conventional settings to adversarial machine learning is shown in Fig. 23.

Spectrum Poisoning Attack Adversarial deep learning can be also used to launch *over-the-air spectrum poisoning attacks* [75]. Using the results of exploratory attack, the adversary falsifies the transmitter's spectrum sensing data over the air by transmitting during transmitter's short spectrum sensing period. Depending on whether the transmitter uses the sensing data as test data to make transmit decisions or for retraining purposes, either it is fooled into making incorrect decisions (evasion attack), or the transmitter's algorithm is retrained incorrectly (causative attack). Both attacks substantially reduce the transmitter's throughput. Note that these attacks differ from the SSDF attack, since the adversary does not participate in cooperative spectrum sensing and does not try to change channel status labels directly. Instead, the adversary injects *adversarial perturbations* to the channel and aims to fool the transmitter into making wrong spectrum access decisions. A defense scheme can be applied by the transmitter that deliberately makes a small number of incorrect transmissions (selected by the confidence score on channel classification) to manipulate the adversary's training data. This defense effectively fools the adversary and helps the transmitter sustain its throughput [75].

Another attack that targets spectrum sensing is *priority violation* attack [77], where the adversary transmits during the sensing phase by pretending to have higher priority (e.g., emulating primary user behavior) and forces a target transmitter into making wrong decisions in an evasion attack.

Evasion Attack Against Signal Classifiers *Adversarial perturbations* can be added to data samples in the test phase for other wireless communications tasks such as signal classification [78–81]. In this *evasion attack*, a receiver aims to classify the

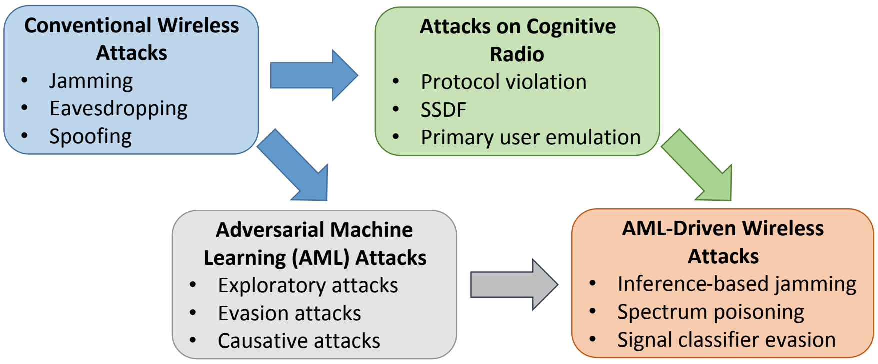

Conventional Wireless Attacks
• Jamming
• Eavesdropping
• Spoofing
Attacks on Cognitive Radio
• Protocol violation
• SSDF
• Primary user emulation
Adversarial Machine Learning (AML) Attacks
• Exploratory attacks
• Evasion attacks
• Causative attacks
AML-Driven Wireless Attacks
• Inference-based jamming
• Spectrum poisoning
• Signal classifier evasion

Fig. 23 From conventional wireless attacks to adversarial machine learning

incoming signals with respect to waveform characteristics. In the meantime, an adversary transmits as well such that a carefully controlled interference signal is added to the received signal and causes the classifier to misclassify the received signal. This problem was studied in [78, 79] for modulation classification using a CNN-based classifier. Both white-box and black-box attacks on the deep learning classifier are shown to be effective in terms of increasing the classification error with small over-the-air perturbations added to the received signal. References [80, 81] developed means to prevent an intruder from successfully identifying the modulation scheme being used.

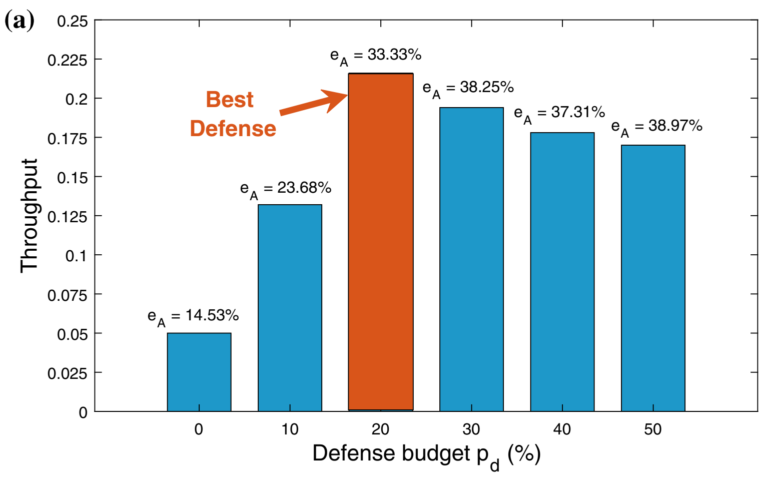
Overall, the attacks that target spectrum sensing or signal classification transmit short signals with low power. Therefore, they are more energy efficient and harder to detect compared to conventional attacks that jam the long data transmission period.

Deep Learning-based Defense Against Wireless Threats In addition to adversarial deep learning, wireless security threats have been studied with defense mechanisms based on deep learning. Against jamming attacks, [82] developed a deep Q-network algorithm for cognitive radios to decide whether to leave an area of heavy jamming or choose a frequency-hopping pattern to defeat smart jammers. Reference [83] trained a CNN network to classify signals to audio jamming, narrowband jamming, pulse jamming, sweep jamming, and spread spectrum jamming. Reference [84] applied a wavelet-based pre-processing step that highlights the disrupted parts of the signal before classifying signals as jammers using a CNN. Another example is *signal authentication* with deep learning as an IoT application. Reference [85] presented a deep learning solution based on a long short-term memory (LSTM) structure to extract a set of stochastic features from signals generated by IoT devices and dynamically watermark these features into the signal. This method was shown to effectively authenticate the reliability of the signals.

4.4 Defense Against Adversarial Deep Learning

A typical first step of adversarial deep learning is the exploratory attack where A builds the surrogate classifier C_A to infer the transmit behavior of T. An effective defense follows from disrupting the training process of C_A. In this defense, T does not always follow the labels returned by C_A and changes them for some of its data samples when making transmit decisions [57]. In particular, T changes the label "ACK" (i.e., "a successful transmission") to "No ACK" (i.e., "no successful transmission"), and vice versa. This way, A's training data is manipulated and A cannot build a reliable classifier in the exploratory attack. As T poisons the training process of A by providing wrong training data, this defense corresponds to a *causative* (or *poisoning*) attack of T back at J. By deliberately taking wrong decisions in certain time slots, T does not transmit even if channel is predicted as idle, and transmits even if channel is predicted as busy.

While this defense increases the uncertainty at A, there is a trade-off in the sense that wrong transmit decisions would reduce the transmission success of T. Therefore, T needs to decide to flip its decision in a small number of carefully selected time slots. Let p_d denote the percentage (%) of time slots in which T decides to flip labels. p_d is considered as a defense budget. T uses the likelihood score $S_T(t)$ (namely the likelihood of whether a channel is idle) returned by DNN to decide when to take the defense action. If $S_T(t)$ is less than a threshold η, T classifies a given time slot t as idle; otherwise T classifies it as busy. When $S_T(t)$ is far away from η, then such a classification has a high confidence; otherwise the confidence is low. For the FNN structure used in previous subsection, $\eta = 0.25$, which is chosen to minimize e_T.



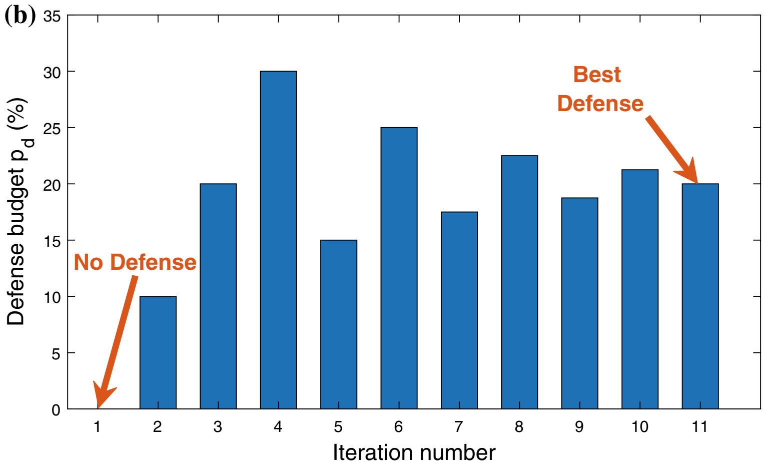


Fig. 24 **a** Effects of transmitter's defense on the adversary [57], **b** Dynamic adaptation of transmitter's defense against the adversary [57]

To optimize the defense mechanism, T performs defense operations in a time slot t when $S_T(t)$ is close to 0 or 1, since T's transmission decisions are more predictable in such a time slot. Subject to p_d values, T changes labels in different time slots and A ends up building different classifiers with different hyperparameters compared to the previous case of no defense.

Figure 24a shows the results when T operates with different defense budgets. As p_d increases, A's error probabilities and T's throughput start increasing significantly. T's throughput reaches maximum when $p_d = 10\%$. As p_d increases further, the growth in A's error probabilities saturates and cannot compensate the errors in channel access decisions anymore. As a result, T's throughput starts decreasing. To determine the best value of p_d, T can start attack mitigation with a fixed level of p_d and then gradually increase or decrease p_d in response to changes in its throughput that is measured through the received ACK messages. Figure 24b shows how p_d is adapted over time to optimize the throughput.

Take-away: This section showed that deep learning can be effectively used in an adversarial setting to launch successful attacks to reduce communication performance. In turn, the adversary can be fooled by manipulating its sensing data samples at certain time instances that are selected by deep learning prediction results.

5 Conclusion

Deep learning has made rapid strides in addressing unique challenges encountered in wireless communications that need to learn from and adapt to spectrum dynamics quickly, reliably, and securely. We presented the recent progress made in applying deep learning to *end-to-end (physical layer) communications*, *spectrum situation awareness*, and *wireless security*. First, we discussed how to formulate transmitter and receiver design at the physical layer as an autoencoder that is constructed as DNNs. We showed that this formulation captures channel impairments effectively and improves performance of single and multiple antenna, and multiuser systems significantly compared to conventional communication systems. Second, we showed that deep learning can help with channel modeling and estimation as well as signal detection and classification when model-based methods fail. The GAN can be applied to reliably capture the complex channel characteristics for the purpose of channel estimation or spectrum data augmentation, while CNNs can improve the signal classification accuracy significantly compared to conventional machine learning techniques. Third, we discussed the application of adversarial deep learning to launch jamming attacks against wireless communications. Starting with an exploratory attack, the adversary can use DNNs to reliably learn the transmit behavior of a target communication system and effectively jam it, whereas a defense mechanism can fool the adversary by poisoning its DNN training process.

The research topics discussed in this chapter illustrated key areas where deep learning can address model and algorithm deficits, enhancing wireless communications. The progress so far clearly demonstrated that deep learning offers new design options for wireless communications and enhances spectrum situational awareness, while adversarial use of deep learning poses an emerging threat to wireless communications and casts communications and sensing into an interesting adversarial game. Numerous additional deep learning applications in wireless communications are on the horizon, which will potentially change the way we model, design, implement, and operate new generations of wireless systems, and shift the field to be more data-centric than ever before.

References

1. Haykin, S.: Cognitive radio: brain-empowered wireless communications. IEEE J. Sel. Areas Commun. **23**(2), 201–220 (2005)
2. Clancy, C., Stuntebeck, H.J., O'Shea, T.: Applications of machine learning to cognitive radio networks. IEEE Trans. Wirel. Commun. **14**(4), 47–52 (2007)
3. O'Shea, T.J., Corgan, J., Charles Clancy, T.: Convolutional radio modulation recognition networks. Int. Conf. Eng. Appl. Neural Netw. (2016)
4. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423, 623–656 (1948)
5. Goldsmith, A.: Joint source/channel coding for wireless channels. IEEE Veh. Technol. Conf. (VTC) (1995)
6. Sagduyu, Y.E., Ephremides, A.: Cross-layer optimization of MAC and network coding in wireless queueing tandem networks. IEEE Trans. Inf. Theory **54**(2), 554–571 (2008)
7. Samuel, N., Diskin, T., Wiesel, A.: Deep MIMO detection. In: IEEE International Workshop on Signal Processing Advances in Wireless Communications (2017)
8. Wang, T., Wen, C., Wang, H., Gao, F., Jiang, T., Jin, S.: Deep learning for wireless physical layer: opportunities and challenges. China Commun. **14**(11), 92–111 (2017)
9. He, H., Wen, C.-K., Jin, S., Li, G.Y.: A model-driven deep learning network for MIMO detection (2018). arXiv preprint arXiv:1809.09336
10. Veeravalli, V.V., Annapureddy, V.S.: Gaussian interference networks: sum capacity in the low interference regime and new outer bounds on the capacity region. IEEE Trans. Inf. Theory **55**(7), 3032–3050 (2009)
11. Han, T.S., Kobayashi, K.: A new achievable rate region for the interference channel. IEEE Trans. Inf. Theory **27**(1), 49–60 (1981)
12. Carleial, A.B.: A case where interference does not reduce capacity. IEEE Trans. Inf. Theory **21**(5), 569–570 (1975)
13. Erpek, T., Ulukus, S., Sagduyu, Y.E.: Interference regime enforcing rate maximization for non-orthogonal multiple access (NOMA). In: IEEE International Conference on Computing, Networking and Communications (ICNC) (2019)
14. O'Shea, T.J., Hoydis, J.: An introduction to deep learning for the physical layer. IEEE Trans. Cogn. Commun. Netw. **3**(4), 563–575 (2017)
15. Dörner, S., Cammerer, S., Hoydis, J., Brink, S.: Deep learning based communication over the air. IEEE J. Sel. Top. Signal Process. **12**(1), 132–143 (2018)
16. O'Shea, T.J., Erpek, T., Charles Clancy, T.: Physical layer deep learning of encodings for the MIMO fading channel. In: IEEE Annual Allerton Conference on Communication, Control, and Computing (Allerton) (2017)
17. Erpek, T., O'Shea, T.J., Charles Clancy, T.: Learning a physical layer scheme for the MIMO interference channel. In: IEEE International Conference on Communications (ICC) (2018)
18. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
19. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). arXiv preprint arXiv:1412.6980

20. Aoudia, F.A., Hoydis, J.: End-to-end learning of communications systems without a channel model (2018). arXiv preprint arXiv:1804.02276
21. Alamouti, S.M.: A simple transmit diversity technique for wireless communications. IEEE J. Sel. Areas Commun. **16**(8), 1451–1458 (1998)
22. Tarokh, V., Seshadri, N., Calderbank, A.R.: Space-time codes for high data rate wireless communication: performance criterion and code construction. IEEE Trans. Inf. Theory **44**(2), 744–765 (1998)
23. Telatar, E.: Capacity of multi-antenna Gaussian channels. Eur. Trans. Telecommun. **10**(6), 585–595 (1999)
24. Yu, W., Rhee, W., Boyd, S., Cioffi, J.M.: Iterative water-filling for Gaussian vector multiple access channels. IEEE Trans. Inf. Theory **50**(1), 145–151 (2004)
25. Tan, X., Xu, W., Be'ery, Y., Zhang, Z., You, X., Zhang, C.: Improving massive MIMO belief propagation detector with deep neural network (2018). arxiv preprint arXiv:1804.01002
26. Liu, L., Oestges, C., Poutanen, J., Haneda, K., Vainikainen, P., Quitin, F., Tufvesson, F., Doncker, P.D.: The COST 2100 MIMO channel model. IEEE Trans. Wirel. Commun. **19**(6), 92–99 (2012)
27. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift (2015). arXiv preprint arXiv:1502.03167
28. Hassibi, B., Hochwald, B.M.: How much training is needed in multiple-antenna wireless links? IEEE Trans. Inf. Theory **49**(4), 951–963 (2003)
29. Erpek, T., Sagduyu, Y.E., Shi, Y., Ponnaluri, S.: Rate optimization with distributed network coordination of multiuser MIMO communications. In: IEEE Vehicular Technology Conference (VTC) (2018)
30. Erpek, T., Sagduyu, Y.E., Shi, Y., Ponnaluri, S.: Network control and rate optimization for multiuser MIMO communications. Ad Hoc Netw. **85**, 92–102 (2019)
31. Chollet, F.: Keras. https://github.com/fchollet/keras (2015)
32. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems. https://tensorflow.org
33. Federal Communications Commission: Notice of proposed rule making and order: facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies, ET Docket No., 03-108 (2005)
34. O'Shea, T.J., Roy, T., West, N., Hilburn, B.C.: Physical layer communications system design over-the-air using adversarial networks (2018). arXiv preprint arXiv:1803.03145
35. Ye, H., Li, G.Y., Juang, B.-H.F., Sivanesan, K.: Channel agnostic end-to-end learning based communication systems with conditional GAN. In: IEEE Global Communications Conference (Globecom) Workshops (2018)
36. O'Shea, T.J., Roy, T., West, N.: Approximating the void: learning stochastic channel models from observation with variational generative adversarial networks (2018). arXiv preprint arXiv:1805.06350
37. Grathwohl, W., Choi, D., Wu, Y., Roeder, G., Duvenaud, D.: Backpropagation through the void: optimizing control variates for black-box gradient estimation (2017). arXiv preprint arXiv:1711.00123
38. Raj, V., Kalyani, S.: Backpropagating through the air: deep learning at physical layer without channel models. IEEE Commun. Lett. **22**(11), 2278–2281 (2018)
39. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (NeurIPS) (2014)
40. Ettus, M.: Universal software radio peripheral (2009)
41. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of Wasserstein GANs. In: Advances in Neural Information Processing Systems (NeurIPS) (2017)
42. O'Shea, T.J., Pemula, L., Batra, D., Charles Clancy, T.: Radio transformer networks: attention models for learning to synchronize in wireless systems (2018). arXiv preprint arXiv:1805.06350
43. O'Shea, T.J., Roy, T., Charles Clancy, T.: Learning robust general radio signal detection using computer vision methods. In: Asilomar Conference on Signals, Systems, and Computers (2017)

44. Akyildiz, I.F., Lee, W.-Y., Vuran, M.C., Mohanty, S.: A survey on spectrum management in cognitive radio networks. IEEE Commun. Mag. **46**(4), 40–48 (2008)
45. O'Shea, T.J., Roy, T., Erpek, T.: Spectral detection and localization of radio events with convolutional neural features. In: European Signal Processing Conference (2017)
46. Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., Batra, D.: Grad-CAM: why did you say that? (2016) arXiv preprint arXiv:1611.07450
47. Lin, M., Chen, Q., Yan, S.: Network in network (2013). arXiv preprint arXiv:1312.4400
48. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: Advances in Neural Information Processing Systems (NeurIPS) (2006)
49. Zeiler, M.D., Krishnan, D., Taylor, G.W., Fergus, R.: Deconvolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
50. O'Shea, T.J., Roy, T., Charles Clancy, T.: Over-the-air deep learning based radio signal classification. IEEE J. Sel. Top. Signal Process. **12**(1), 168–179 (2018)
51. Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R.: Learning from simulated and unsupervised images through adversarial training. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
52. Shi, Y., Sagduyu, Y.E., Davaslioglu, K., Li, J.: Generative adversarial networks for black-box API attacks with limited training data. In: IEEE Symposium on Signal Processing and Information Technology (2018)
53. Shi, Y., Sagduyu, Y.E., Davaslioglu, K., Levy, R.: Vulnerability detection and analysis in adversarial deep learning. Guide to Vulnerability Analysis for Computer Networks and Systems—An Artificial Intelligence Approach. Computer Communications and Networks. Springer, Cham (2018)
54. Mirza, M., Osindero, S.: Conditional generative adversarial nets (2014). arXiv preprint arXiv:1411.1784
55. Davaslioglu, K., Sagduyu, Y.E.: Generative adversarial learning for spectrum sensing. In: IEEE International Conference on Communication (ICC) (2018)
56. Tang, B., Tu, Y., Zhang, Z., Lin, Y.: Digital signal modulation classification with data augmentation using generative adversarial nets in cognitive radio networks. IEEE Access. **6**, 15713–15722 (2018)
57. Erpek, T., Sagduyu, Y.E., Shi, Y.: Deep learning for launching and mitigating wireless jamming attacks. IEEE Trans. Cogn. Commun. Netw. **5**(1), 2–14 (2019)
58. Shi, Y., Sagduyu, Y.E., Erpek, T., Davaslioglu, K., Lu, Z., Li, J.: Adversarial deep learning for cognitive radio security: jamming attack and defense strategies. In: IEEE ICC 2018 Workshop—Promises and Challenges of Machine Learning in Communications Networks (2018)
59. Hughes, B., Bothe, S., Farooq, H., Imran, A.: Generative adversarial learning for machine learning empowered self organizing 5G networks. In: IEEE International Conference on Computing, Networking and Communications (ICNC). http://bsonlab.com/images/Publications/Conferences/C18-4.pdf (2019)
60. Donahue, J., Krahenbuhl, P., Darrell, T.: Adversarial feature learning (2016). arXiv preprint arXiv:1605.09782
61. Shi, Y., Davaslioglu, K., Sagduyu, Y.E.: Generative adversarial network for wireless signal spoofing. In: ACM WiSec Workshop on Wireless Security and Machine Learning (WiseML) (2019)
62. Sagduyu, Y.E., Berry, R., Ephremides, A.: Jamming games in wireless networks with incomplete information. IEEE Commun. Mag. **49**(8), 112–118 (2011)
63. Sagduyu, Y.E., Ephremides, A.: A game-theoretic analysis of denial of service attacks in wireless random access. J. Wirel. Netw. **15**(5), 651–666 (2009)
64. Sagduyu, Y.E., Berry, R., Ephremides, A.: Jamming games for power controlled medium access with dynamic traffic. In: IEEE International Symposium on Information Theory (ISIT) (2010)
65. Sagduyu, Y.E., Berry, R., Ephremides, A.: Wireless jamming attacks under dynamic traffic uncertainty. In: IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WIOPT) (2010)

66. Sagduyu, Y.E., Berry, R., Ephremides, A.: MAC games for distributed wireless network security with incomplete information of selfish and malicious user types. In: IEEE International Conference on Game Theory for Networks (GameNets) (2009)
67. Sagduyu, Y.E.: Securing cognitive radio networks with dynamic trust against spectrum sensing data falsification. In: IEEE Military Communications Conference (MILCOM) (2014)
68. Lu, Z., Sagduyu, Y.E., Li, J.: Securing the backpressure algorithm for wireless networks. IEEE Trans. Mobile Comput. **16**(4), 1136–1148 (2017)
69. Lu, Z., Sagduyu, Y.E., Li, J.: Queuing the trust: secure backpressure algorithm against insider threats in wireless networks. In: IEEE Conference on Computer Communications (INFOCOM) (2015)
70. Vorobeychik, Y., Kantarcioglu, M.: Adversarial Machine Learning. Morgan & Claypool (2018)
71. Shi, Y., Sagduyu, Y.E., Grushin, A.: How to steal a machine learning classifier with deep learning. In: IEEE Symposium on Technologies for Homeland Security (2017)
72. Shi, Y., Sagduyu, Y.E.: Evasion and causative attacks with adversarial deep learning. In: IEEE Military Communications Conference (MILCOM) (2017)
73. Pi, L., Lu, Z., Sagduyu, Y.E., Chen, S.: Defending active learning against adversarial inputs in automated document classification. In: IEEE Global Conference on Signal and Information Processing (GlobalSIP) Symposium on Compressed Sensing, Deep Learning (2016)
74. Shi, Y., Sagduyu, Y.E., Davaslioglu, K., Li, J.: Active deep learning attacks under strict rate limitations for online API calls. In: IEEE Symposium Technologies for Homeland Security (2018)
75. Shi, Y., Erpek, T., Sagduyu, Y.E., Li, J.: Spectrum data poisoning with adversarial deep learning. In: IEEE Military Communications Conference (MILCOM) (2018)
76. Microsoft Cognitive Toolkit (CNTK), https://docs.microsoft.com/en-us/cognitive-toolkit
77. Sagduyu, Y.E., Shi, Y., Erpek, T.: IoT network security from the perspective of adversarial deep learning. In: IEEE International Conference on Sensing, Communication and Networking (SECON) Workshop on Machine Learning for Communication and Networking in IoT (2019)
78. Sadeghi, M., Larsson, E.G.: Adversarial attacks on deep-learning based radio signal classification. IEEE Wirel. Commun. Lett. **8**(1), 213–216 (2019)
79. Flowers, B., Buehrer, R.M., Headley, W.C.: Evaluating adversarial evasion attacks in the context of wireless communications (2019). arXiv preprint arXiv:1903.01563
80. Hameed, M.Z., Gyorgy, A., Gunduz, D.: Communication without interception: defense against deep-learning-based modulation detection (2019). arXiv preprint arXiv:1902.10674
81. Kokalj-Filipovic, S., Miller, R.: Adversarial examples in RF deep learning: detection of the attack and its physical robustness (2019). arXiv preprint arXiv:1902.06044
82. Han, G., Xiao, L., Poor, H.V.: Two-dimensional anti-jamming communication based on deep reinforcement learning. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2017)
83. Wu, Z., Zhao, Y., Yin, Z., Luo, H.: Jamming signals classification using convolutional neural network. In: IEEE Symposium on Signal Processing and Information Technology (ISSPIT) (2017)
84. Topal, O.A., Gecgel, S., Eksioglu, E.M., Kurt, G.: Identification of smart jammers: learning based approaches using wavelet representation (2019). arXiv preprint arXiv:1901.09424
85. Ferdowsi, A., Saad, W.: Deep learning for signal authentication and security in massive internet of things systems (2018). arXiv preprint arXiv:1803.00916

Identifying Extremism in Text Using Deep Learning

Andrew Johnston and Angjelo Marku

Abstract Various forms of terrorism have become increasingly relevant in today's world. Consequently, the utilization of the web by various terrorist groups to spread propaganda, communicate and organize has increased. However, techniques to effectively identify such material are lacking. This chapter explores an approach which can classify any piece of text as belonging to one of four extremist groups: Sunni Islamic, Antifascist Groups, White Nationalists and Sovereign Citizens. This classification is performed by LSTM models, which will be proven to be much more effective than non-deep learning approaches. This chapter will describe the performance of various models in detail. The process of creating good quality datasets for each extremist category and the unique challenges such a task presents will also be explored.

Keywords Deep learning · Terrorism · LSTM · Extremism · Text classification · Logistic regression

1 Introduction

Terrorism has been on the rise within the United States in the past decade [1]. Moreover, groups like ISIL and al-Qaeda have mounted aggressive online campaigns to identify disenfranchised youth in Western countries and encourage them to perform acts of terror or migrate to hot-zones within the Middle East. These recruitment messages often come in the form of visually-appealing propaganda, such as the Dabiq and Rumiyah, magazines published by the ISIL in a variety of languages. These magazines contain messages justifying the terrorist cause as well as giving in-depth explanations of how to build destructive devices. Inspire, a similar publication run

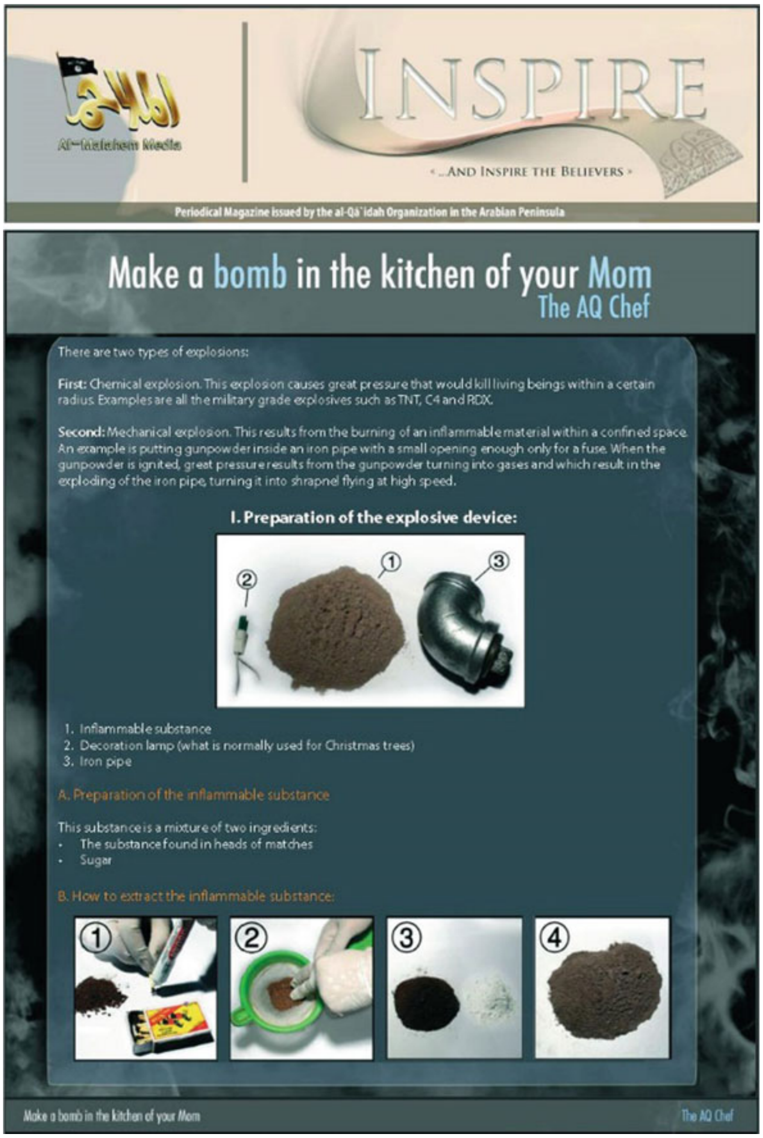
A. Johnston (✉) · A. Marku (✉)
Department of Computer Science, Graduate School of Arts and Science, Fordham University, New York, NY, USA
e-mail: ajohnston9@fordham.edu

A. Marku
e-mail: amarku2@fordham.edu

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5_10

by al-Qaeda, created the infamous article “Make a bomb in the kitchen of you Mom” as shown in Fig. 1.

International terror organizations are following the lead of many domestic terror organizations that utilize the Internet’s affordability and reach to identify and recruit members. Websites such as Stormfront, a website dedicated to Neo-Nazi and White


الملاحم
Al-Malahem Media
INSPIRE
«...AND INSPIRE THE BELIEVERS»
Periodical Magazine issued by the al-Qā`idah Organization in the Arabian Peninsula

Make a bomb in the kitchen of your Mom
The AQ Chef

There are two types of explosions:

First: Chemical explosion. This explosion causes great pressure that would kill living beings within a certain radius. Examples are all the military grade explosives such as TNT, C4 and RDX.

Second: Mechanical explosion. This results from the burning of an inflammable material within a confined space. An example is putting gunpowder inside an iron pipe with a small opening enough only for a fuse. When the gunpowder is ignited, great pressure results from the gunpowder turning into gases and which result in the exploding of the iron pipe, turning it into shrapnel flying at high speed.

I. Preparation of the explosive device:

1. Inflammable substance
2. Decoration lamp (what is normally used for Christmas trees)
3. Iron pipe

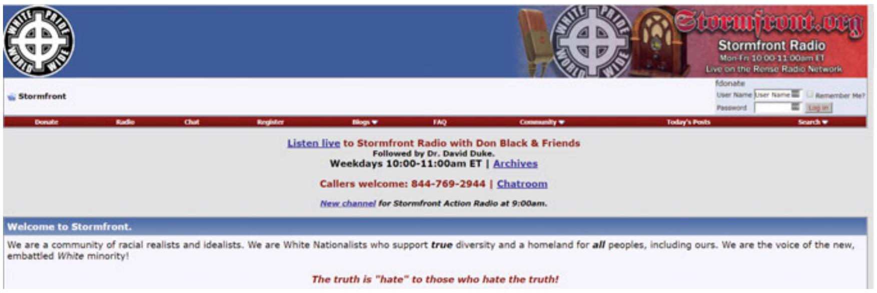
A. Preparation of the inflammable substance

This substance is a mixture of two ingredients:
- The substance found in heads of matches
- Sugar

B. How to extract the inflammable substance:

Make a bomb in the kitchen of your Mom The AQ Chef

Fig. 1 An inspire article describing how to make improvised explosive devices (IEDs)



Listen live to Stormfront Radio with Don Black & Friends
Followed by Dr. David Duke.
Weekdays 10:00-11:00am ET | Archives

Callers welcome: 844-769-2944 | Chatroom

New channel for Stormfront Action Radio at 9:00am.

Welcome to Stormfront.

We are a community of racial realists and idealists. We are White Nationalists who support ***true*** diversity and a homeland for ***all*** peoples, including ours. We are the voice of the new, embattled *White* minority!

The truth is "hate" to those who hate the truth!

Fig. 2 The main page of the Stormfront website, a Neo-Nazi and White Supremacist organization

Supremacist ideologies since November 1996 [2] are increasingly used for this purpose. Figure 2 shows the main page of Stormfront. Its ideology is described using the term “race realism,” a phrase utilized by White Supremacists.

Although sites like Stormfront and publications such as Dabiq, Rumiyah, and Inspire are well-known, terror recruiting has also taken to more mainstream social media sites. Tumblr, a popular blogging platform, has become a favorite among ISIS to recruit Westerners to the cause [3] as shown in Fig. 3.

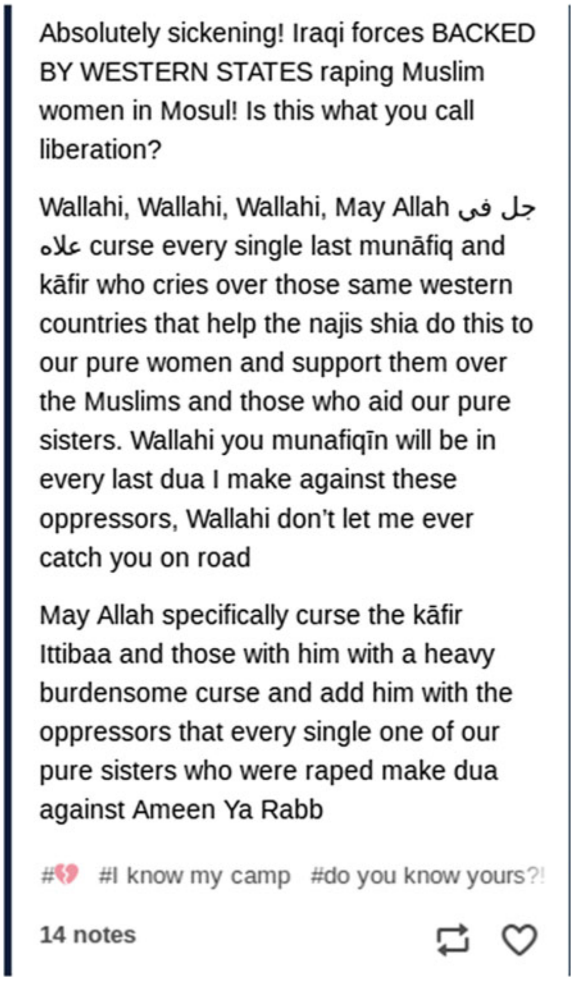
Similarly, Twitter has become a favorite platform for ISIS members to promote propaganda. Although many of the Twitter accounts eventually become suspended, preliminary investigation suggests that there may be over 170,000 active ISIS related accounts on Twitter [4].

For members of law enforcement and intelligence communities, finding, investigating, and stopping online terrorism is a significant challenge. Given the massive size of social media networks, scrutinizing every post would be a monumental task. Consequently, such professionals must choose between a variety of sub-par methods to find new terrorist-created content.

The first strategy is to identify a “lead”, which is a known account or a post that promotes terrorism. Using information derived from the post such as interactions with the post (replies, retweets, likes, or favorites), a professional can seek to identify related accounts and begin mapping out a network of potentially suspicious accounts for further investigation. The principal problem with such a method is that it is limited to the network of users identifiable through investigating the principal lead. If a separate network of terrorists has no interaction with the network of the initial target, they likely will not be identified through this method.

The second method involves using search engines and similar tools (such as a Twitter search) to identify posts of interest using keywords. The Department of Homeland Security (DHS) created the Analyst’s Desktop Binder that details words and phrases related to terrorism and similar security concerns [5]. An excerpt of potential search terms in the Binder is shown in Fig. 4.

Although terrorists could potentially be identified using these terms, such an approach is fundamentally limited. In order to maintain usefulness, such a list would have to be regularly updated to include new terms and techniques as well as casual or


Absolutely sickening! Iraqi forces BACKED BY WESTERN STATES raping Muslim women in Mosul! Is this what you call liberation?

Wallahi, Wallahi, Wallahi, May Allah جل فى علاه curse every single last munāfiq and kāfir who cries over those same western countries that help the najis shia do this to our pure women and support them over the Muslims and those who aid our pure sisters. Wallahi you munafiqīn will be in every last dua I make against these oppressors, Wallahi don't let me ever catch you on road

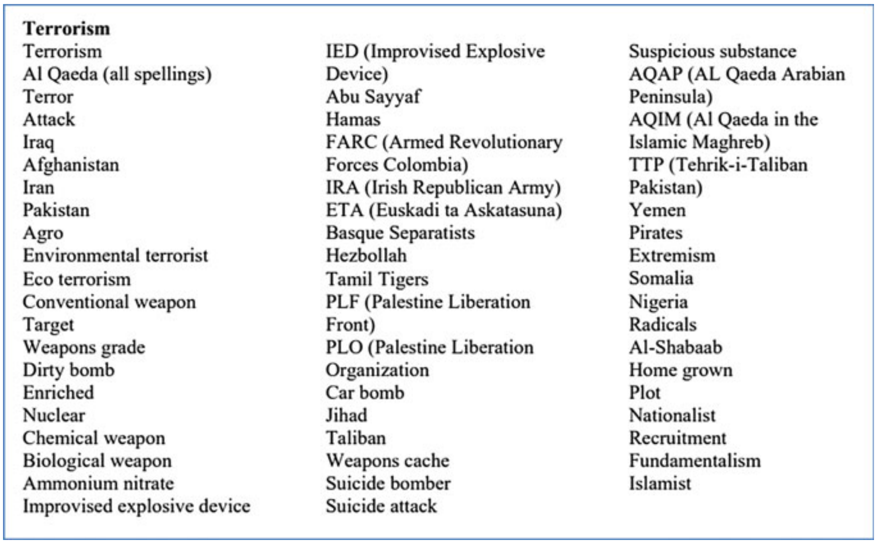
May Allah specifically curse the kāfir Ittibaa and those with him with a heavy burdensome curse and add him with the oppressors that every single one of our pure sisters who were raped make dua against Ameen Ya Rabb

#💔 #I know my camp #do you know yours?!

14 notes

Fig. 3 A ISIS-affiliated terrorist posting on Tumblr

slang terms. Moreover, such a list would have to include translations of terms as well as various transliterations. Even with a more comprehensive list, this strategy is still fallible to obsfucation techniques such as the usage of homoglyphs. For example, a terrorist wishing to evade detection through searching might replace the word "jihad" with "j1had". Although the substitution of the number one for the letter "I" would likely not confuse English readers, it would evade plain searches for the term jihad. Moreover, searching as a principal technique will yield a significant amount of false positives, such as news stories or tangential posts. For instance, the term "plot" referenced in Fig. 4 could appear in a post describing a "terrorist plot" as well as a post about the plot of a movie or even a typographical error from a user attempting to type "pot" in a post that otherwise would be considered entirely benign.



Terrorism

- Terrorism
- Al Qaeda (all spellings)
- Terror
- Attack
- Iraq
- Afghanistan
- Iran
- Pakistan
- Agro
- Environmental terrorist
- Eco terrorism
- Conventional weapon
- Target
- Weapons grade
- Dirty bomb
- Enriched
- Nuclear
- Chemical weapon
- Biological weapon
- Ammonium nitrate
- Improvised explosive device
- IED (Improvised Explosive Device)
- Abu Sayyaf
- Hamas
- FARC (Armed Revolutionary Forces Colombia)
- IRA (Irish Republican Army)
- ETA (Euskadi ta Askatasuna)
- Basque Separatists
- Hezbollah
- Tamil Tigers
- PLF (Palestine Liberation Front)
- PLO (Palestine Liberation Organization)
- Car bomb
- Jihad
- Taliban
- Weapons cache
- Suicide bomber
- Suicide attack
- Suspicious substance
- AQAP (AL Qaeda Arabian Peninsula)
- AQIM (Al Qaeda in the Islamic Maghreb)
- TTP (Tehrik-i-Taliban Pakistan)
- Yemen
- Pirates
- Extremism
- Somalia
- Nigeria
- Radicals
- Al-Shabaab
- Home grown
- Plot
- Nationalist
- Recruitment
- Fundamentalism
- Islamist

Fig. 4 Potential search terms proposed by the DHS for identifying terrorist propaganda

In this chapter, we describe the process and results of creating models to detect different types of extremism. With a focus on Sunni, radical leftist ("Antifa"), White Supremacist, and Sovereign Citizen extremism, we describe the process of creating datasets and building models that can identify terrorist propaganda and analyze their ability to withstand techniques such as obfuscation with homoglyphs. We also analyze public datasets and justify the need for the creation of larger, more comprehensive datasets for counterterrorism research. With our results, we demonstrate the necessity of deep learning approaches to building such models and describe the applications such models could have in the modern fight against domestic and foreign terror.

2 Background

The motive for terrorism varies wildly depending on the underlying methodology. The difference in motive often has a significant impact on terminology and the structure of propaganda that must be taken into account when developing models to identify such material. In this section, we will give a brief overview of the terrorist ideologies we are investigating and their propaganda techniques.

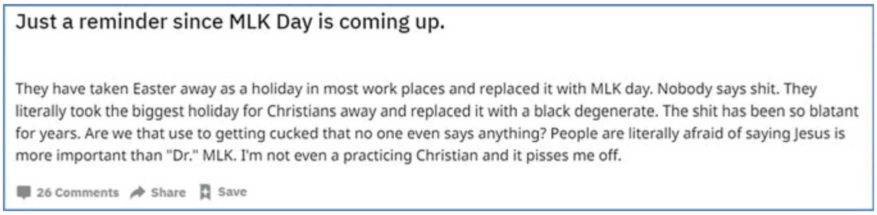
Sunni extremism, the parent ideology of well-known terror groups such as al-Qaeda and ISIL, is centered on the establishment of an Islamic government (a "caliphate") and the dismantlement of governments deemed incompatible with Islamic ideology, such as the majority of Western governments. Groups such as

ISIL have established themselves as archetypal Islamic government and consider themselves an independent state [6]. Terrorism is often referred to as “jihad” and is considered by Sunni extremist groups to be a defensive action [7]. Moreover, Sunni extremism is notable for the justification of suicide-based attacks where the attacker mounts an attack that will likely kill themselves in the process, such as suicide bombing. This justification is established through Sunni extremism groups issuing proclamations (“fatwas”) that such suicide attacks are considered to be martyrdom for the cause rather than suicide, which is otherwise impermissible [8]. Consequently, propaganda often glorifies such suicide operations, with groups such as ISIS publicly condoning such operations in the global media [9].

White Supremacy is centered on the belief that non-white races are principally responsible for societal problems and crime. White Supremacy is also blended with anti-Semitic and Neo-Nazi ideologies. Figure 5 shows a Reddit posting consistent with White Supremacist ideology.

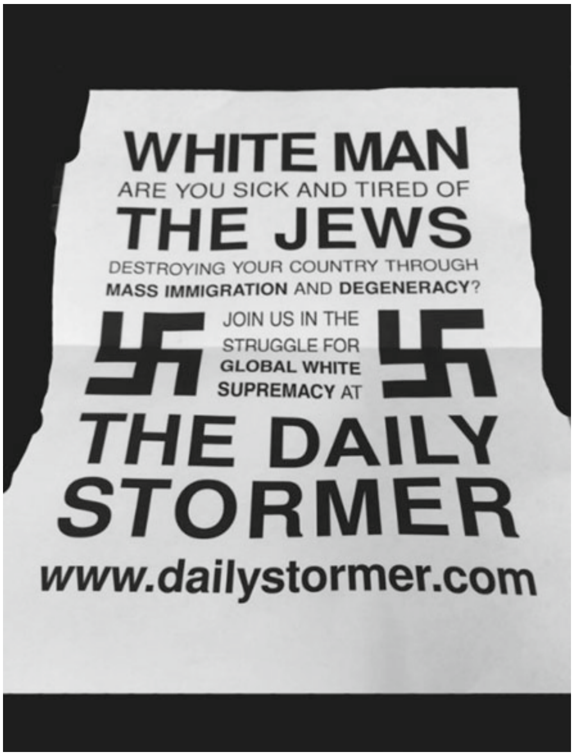
White Supremacists are responsible for violence, but often lack the group cohesion of other terrorist groups, instead committing violence in “lone wolf” attacks. For example, James Jackson murdered an African American in hopes of the attack provoking a “race war”, a concept in White Supremacist ideology where interracial conflict is strong enough to provoke continued violence [10]. Although historically-relevant White Supremacist groups such as the Ku Klux Klan (KKK) still operate within the United States, their numbers are decreasing with current membership numbers estimated to be around 5,000 to 8,000 [11]. Likewise, KKK groups have dwindled due to internal conflict and successful government operations [11]. Neo-Nazi groups often adopt White Supremacist language but have a more specified intent of re-establishing National Socialism as prescribed by the German Nazi party of the Second World War. Figure 6 demonstrates the close ties between White Supremacist and Neo-Nazi ideologies. The flyer contains both overt references to Nazism, such as the swastika, as well as references to topics common in White Supremacist propaganda, such as “mass immigration” and “degeneracy”. In White Supremacist propaganda, “degeneracy” is the belief that modern culture has led to a loss of ethics and traditional values and will lead to the decline of civilization.

Antifascist groups, commonly referred to as “Antifa”, have recently been identified as terrorist groups within the United States. Antifa ideology claims to be focused on preventing growth of Neo-Nazi and White Supremacist groups, but Antifascists


Just a reminder since MLK Day is coming up.

They have taken Easter away as a holiday in most work places and replaced it with MLK day. Nobody says shit. They literally took the biggest holiday for Christians away and replaced it with a black degenerate. The shit has been so blatant for years. Are we that use to getting cucked that no one even says anything? People are literally afraid of saying Jesus is more important than "Dr." MLK. I'm not even a practicing Christian and it pisses me off.

Fig. 5 A Reddit posting from a subforum dedicated to White Supremacy



WHITE MAN
ARE YOU SICK AND TIRED OF
THE JEWS
DESTROYING YOUR COUNTRY THROUGH
MASS IMMIGRATION AND DEGENERACY?
JOIN US IN THE
STRUGGLE FOR
GLOBAL WHITE
SUPREMACY AT
THE DAILY
STORMER
www.dailystormer.com

Fig. 6 A propaganda flyer demonstrating the link between Neo-Nazi and White Supremacist ideologies

have often attacked political conservatives and police officers, claiming such victims are secretly in favor of Neo Nazi or White Supremacist ideologies. Moreover, Antifa have been known to attack journalists covering Antifa protests without provocation from the journalists [12]. Conservative journalist Steven Crowder, host of the podcast Louder with Crowder, captured footage while undercover with Antifa during their preparations to protest and attack attendees of a talk given by notable Jewish Conservative Ben Shapiro, who they considered to be a Neo-Nazi [13]. Their investigation demonstrated a willingness of the Antifa group to provoke armed conflict, using weapons such as ice picks, combat knives, semi-automatic rifles, and illegally modified shotguns [13]. Overtly, many Antifa organizations characterizes their violence as limited to defense from attacks from Neo-Nazi terrorists, often using the phrase "Bash the Fash" (referring to Fascists) in public propaganda, as shown in Fig. 7.

Critical to Antifa philosophy is the concept of "dog whistles", which are allegedly covert references to White Supremacist or Neo-Nazi ideologies intended to signal a person's true belief. One such claimed dog whistle is the "OK" hand sign made by raising three fingers while holding the thumb and index finger together in a circle. The sign purported is a reference to the phrase "White Power" due to the fingers making the approximate shapes of a "W" and "P". One White House intern was

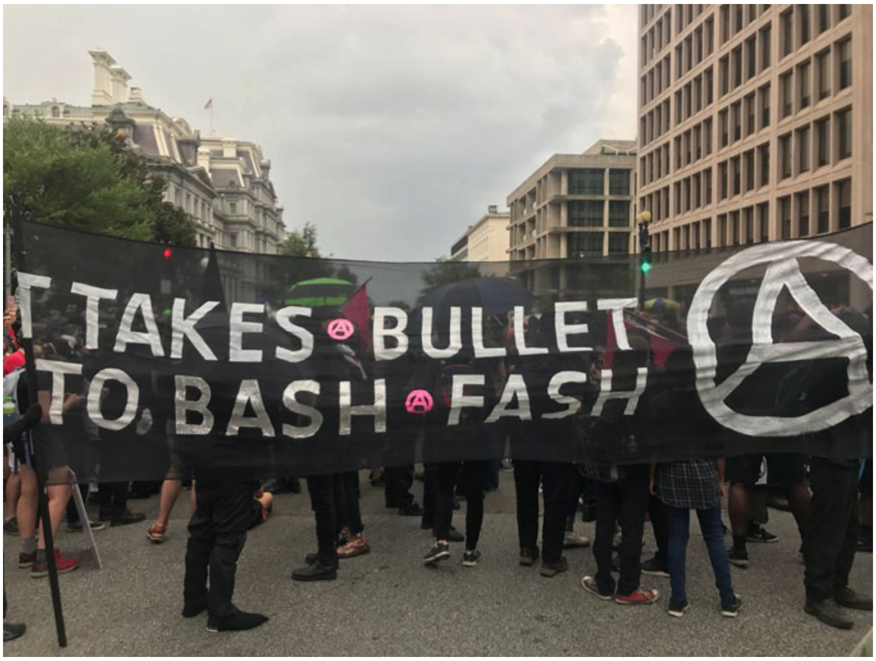


Fig. 7 An Antifa protest sign using the phrase "Bash the Fash", a common Antifa call to violence

labelled as a Nazi by Antifa groups for allegedly utilizing the dog whistle in a White House photo, shown in Fig. 8.

Sovereign Citizen terrorist groups, also sometimes referred to as "Freemen on the Land," are groups purporting to be exempt from the majority of United States laws due to a difference in interpretation of law. Sovereign Citizens are known for encounters

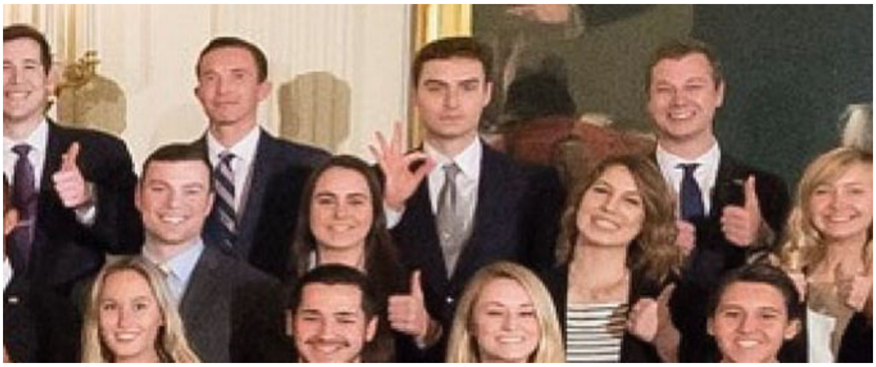
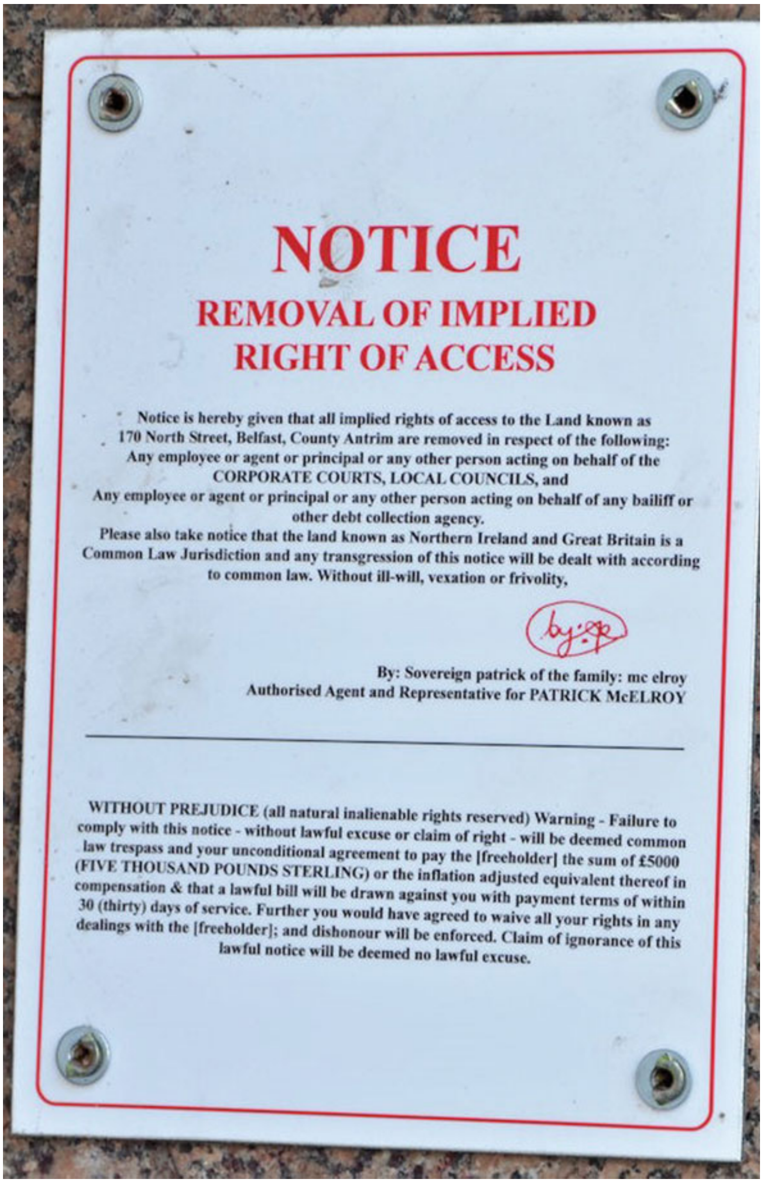


Fig. 8 Antifa groups claimed the White House intern was using a "dog whistle" to indicate White Supremacist beliefs



NOTICE

REMOVAL OF IMPLIED RIGHT OF ACCESS

Notice is hereby given that all implied rights of access to the Land known as 170 North Street, Belfast, County Antrim are removed in respect of the following:
Any employee or agent or principal or any other person acting on behalf of the CORPORATE COURTS, LOCAL COUNCILS, and
Any employee or agent or principal or any other person acting on behalf of any bailiff or other debt collection agency.
Please also take notice that the land known as Northern Ireland and Great Britain is a Common Law Jurisdiction and any transgression of this notice will be dealt with according to common law. Without ill-will, vexation or frivolity,

[illegible]

By: Sovereign patrick of the family: mc elroy
Authorised Agent and Representative for PATRICK McELROY

WITHOUT PREJUDICE (all natural inalienable rights reserved) Warning - Failure to comply with this notice - without lawful excuse or claim of right - will be deemed common law trespass and your unconditional agreement to pay the [freeholder] the sum of £5000 (FIVE THOUSAND POUNDS STERLING) or the inflation adjusted equivalent thereof in compensation & that a lawful bill will be drawn against you with payment terms of within 30 (thirty) days of service. Further you would have agreed to waive all your rights in any dealings with the [freeholder]; and dishonour will be enforced. Claim of ignorance of this lawful notice will be deemed no lawful excuse.

Fig. 9 An Irish Sovereign Citizen sign

with law enforcement while utilizing fraudulent documents, such as license plates, driver's licenses, and currency [14]. These encounters are more frequently turning violent, resulting in the death of police officers who believed they were performing routine traffic stops; such occurrences have warranted the Florida Sherriff's Department to issue specialized training for handling suspects that are potentially Sovereign Citizens [15]. A key belief is that the legal name of a citizen is used by the government to ensnare unknowing people (i.e. everyone except Sovereign Citizens) into performing civic duties that Sovereign Citizens believe are optional, such as paying taxes, purchasing various licenses, or receiving punishment after being judged guilty in a court of law [16]. Consequently, Sovereign Citizens will use a combination of various legal terms when referring to themselves, such as a person named "John Doe" referring to themselves as "John Doe, Executive Trustee for the Private Contract Trust known as JOHN DOE" [16]. Sovereign Citizens are known to be more organized than other domestic terror groups, and have demonstrated interest in armed takeovers, such as the 2014 takeover of the Malheur National Wildlife Refuge Headquarters, where Sovereign Citizens occupied a federal building to demand that the federal government return land to the state governments. Although the standoff ended with only one casualty and one injury (both Sovereign Citizens), such events could embolden future, more lethal events. Enabling the spread of the ideology is the easily obtainable fake documents and signs, such as the one from an Irish Sovereign Citizen group shown in Fig. 9.

Although terror groups may vary in motive and intentions in their ideology, the core tenants are the same: they wish to do harm to those that do not subscribe to their ideology and strike fear into society at large. Moreover, the increasing prevalence of such terror groups indicate that traditional methods of identification and dismantling are failing to counter the threat posed by internet-enabled terror groups.

3 Dataset

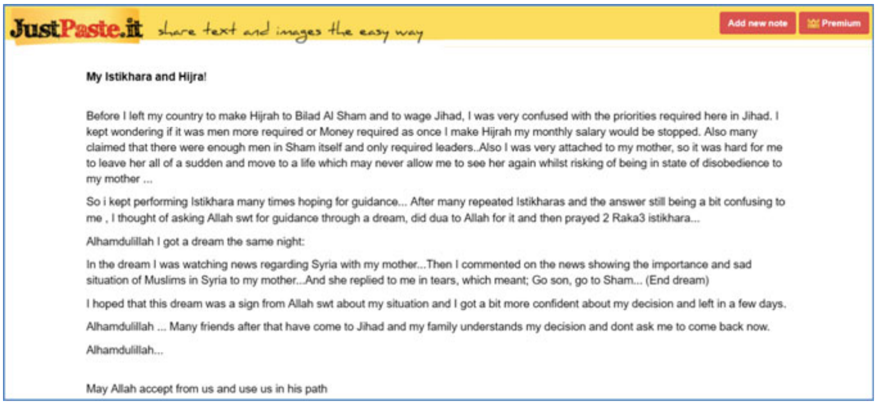
The primary challenge of creating models to identify terrorist propaganda is assembling appropriate datasets. Given the covert nature of such groups, identifying material to use can be challenging. Moreover, many social media sites will quick block and suspend material as well as accounts known to promote such material, making leads on the location of such material expire quickly.

In order to utilize most deep learning methods, a binary dataset has to be established. While identifying material for the extremist label is relatively straightforward, identifying relevant data for the benign label is more challenging. Arbitrary benign data such as web pages on miscellaneous topics can serve some value, but the benign dataset also needs curated texts to prevent common sources of false positives. These curated texts can include news articles, religious texts, and encyclopedia entries. Such curation can prevent common sources of error such as articles discussing events involving extremism that lack support for the causes being discussed. Likewise, the inclusion of religious texts in benign datasets can prevent errors resulting from the quotation of religious literature being mistaken for religiously motivated types of extremism such as Sunni extremism or White Nationalism. Although extremist propaganda may include religious quotations or references, the presence of such is not sufficient to identify material as extremist.

In this section, strategies to compile datasets for different forms of extremism will be discussed. These datasets will be utilized in later sections to develop models, and the importance of inclusion of certain curated materials for the benign class will be justified. Moreover, certain examples from each class will be highlighted to demonstrate the insufficiency of traditional text mining models for the task.

3.1 *Sunni Extremism*

The Sunni extremism dataset compiled in "Identifying Sunni Extremism" by Johnston and Weiss [17] utilized a team of 40 researchers, with experience in intelligence gathering and Middle Eastern culture, Islam, as well as fluency in multiple dialects of Arabic and Farsi. Using compiled intelligence reports from sources such as SiTE Intelligence Group [18], sites hosting known terrorism material such as JustPasteIt [19] and AddPostIt [20] were identified. These sites are "paste" sites, allowing users to upload arbitrary content such as text and images to a unique URL that can be shared. A technique to identify all valid URLs was identified, and the posts were scraped. The researchers separated those promoting Sunni extremism from "benign" posts of unspecified topics. During the categorization, no materials from other terror groups were identified. Benign posts did contain material promoting criminality (e.g. cybercrime, drug trafficking, etc.) that was deemed lacking in any demonstrable links to terrorist propaganda. Figure 10 shows a post from JustPasteIt describing the writer's hijra (journey, in this case to ISIL-controlled territory) for the purpose of committing terrorism (i.e. "waging jihad").



My Istikhara and Hijra!

Before I left my country to make Hijrah to Bilad Al Sham and to wage Jihad, I was very confused with the priorities required here in Jihad. I kept wondering if it was men more required or Money required as once I make Hijrah my monthly salary would be stopped. Also many claimed that there were enough men in Sham itself and only required leaders..Also I was very attached to my mother, so it was hard for me to leave her all of a sudden and move to a life which may never allow me to see her again whilst risking of being in state of disobedience to my mother ...

So i kept performing Istikhara many times hoping for guidance... After many repeated Istikharas and the answer still being a bit confusing to me , I thought of asking Allah swt for guidance through a dream, did dua to Allah for it and then prayed 2 Raka3 istikhara...

Alhamdulillah I got a dream the same night:

In the dream I was watching news regarding Syria with my mother...Then I commented on the news showing the importance and sad situation of Muslims in Syria to my mother...And she replied to me in tears, which meant; Go son, go to Sham... (End dream)

I hoped that this dream was a sign from Allah swt about my situation and I got a bit more confident about my decision and left in a few days.

Alhamdulillah ... Many friends after that have come to Jihad and my family understands my decision and dont ask me to come back now.

Alhamdulillah...

May Allah accept from us and use us in his path

Fig. 10 A post from JustPasteIt promoting joining ISIL

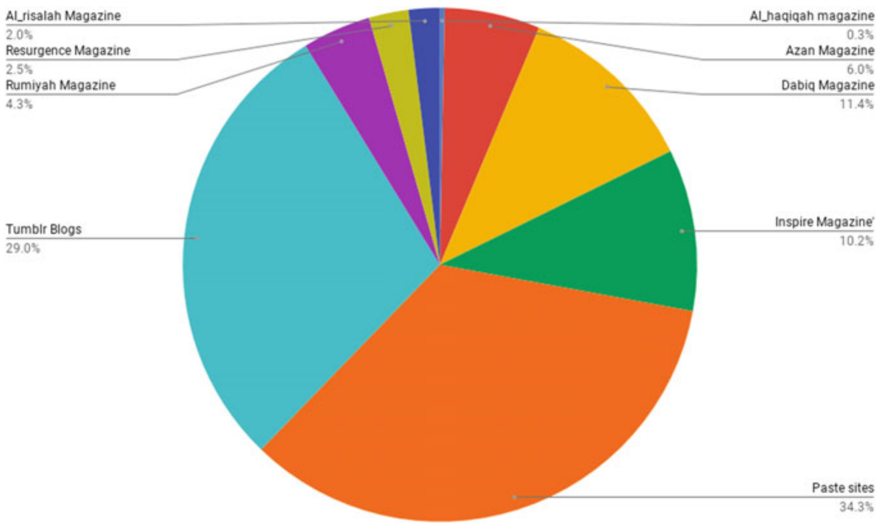


Fig. 11 The breakdown of data for the Sunni dataset

To add additional substance, the extremist dataset was supplemented with issues of the Dabiq, Rumiyah, and Inspire magazines in English as curated by the Clarion Project [21]. These magazines are often over one hundred pages in length, containing high-definition full-quality images and numerous articles. For the purposes of the dataset, the text was copied from the articles, with content such as page numbers removed.

For this chapter, a set of Tumblr posts containing extremism were identified and added to the extremist label using the same team as utilized in the original Johnston and Weiss paper [17]. These posts were identified via an initial lead from JustPasteIt where a single Tumblr blog was linked. Investigations into this user yielded additional Tumblr users posting extremist Sunni content that were also parsed and added to the dataset.

The breakdown of data by source is shown in Fig. 11, where the size of each dataset is counted in words.

3.2 White Nationalism

Given the identified "lone wolf" nature of White Supremacists, identifying suitable content was difficult. Parsers were written to extract posts from Stormfront [22]. A similar forum, known as the Vanguard News Network (VNN) Forum [23], was identified and postings were scraped. Moreover, investigation identified the American Freedom Party, formerly known as the American Third Position Party [24]. The American Third Position Party promotes White Supremacism as a mainstay of their

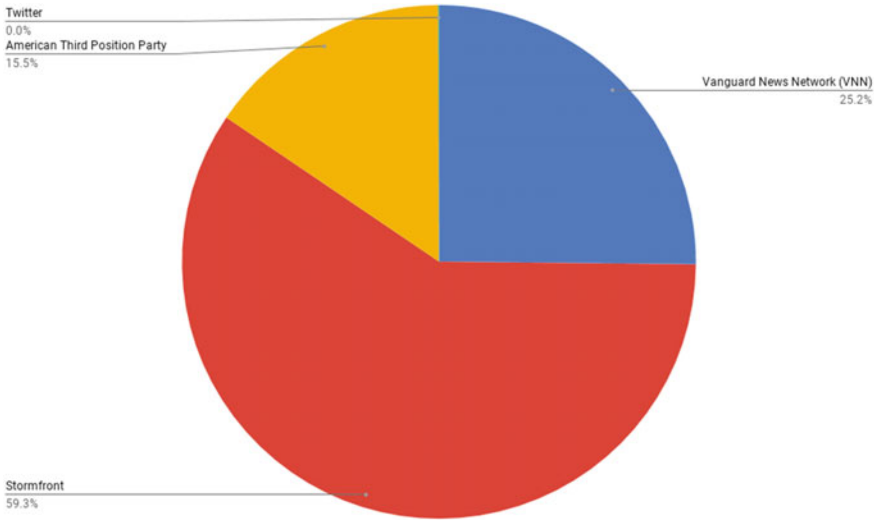


Fig. 12 The breakdown of data for the White Supremacist dataset

political ideology. Content from the “Featured Articles” section of their website, which is dedicated to providing White Supremacist perspective on current events, was scraped and added to the dataset.

Likewise, an existing dataset consisting of hate speech Twitter posts [25] was identified and included. The original dataset was manually reviewed to remove content irrelevant to White Supremacy.

The breakdown of the data for the White Supremacist dataset is shown in Fig. 12. Note that while Twitter data was included, its small size compared to the other sources prevents the graph from displaying the actual percentage of data from Twitter.

3.3 *Antifa*

Since many Antifa groups are regionally focused within the United States, few central forums with large populations of active users were identified. Instead, data from multiple smaller, regional Antifa groups were supplemented with data from a singular large Antifa forum.

NYCAntifa [26] is a group focused on Antifa activities within the New York City region. While this site lacks a forum, it contains a number of pages published by the group promoting Antifa activities and giving detailed information in regards to their position and prospective members. Similarly, the Nomattimen [27], Antifacist Network [28], London Antifacists [29] and Malatesta [30] sites were similarly structured (i.e. blog postings without substantial user interaction) but contained a large amount of content in article format.

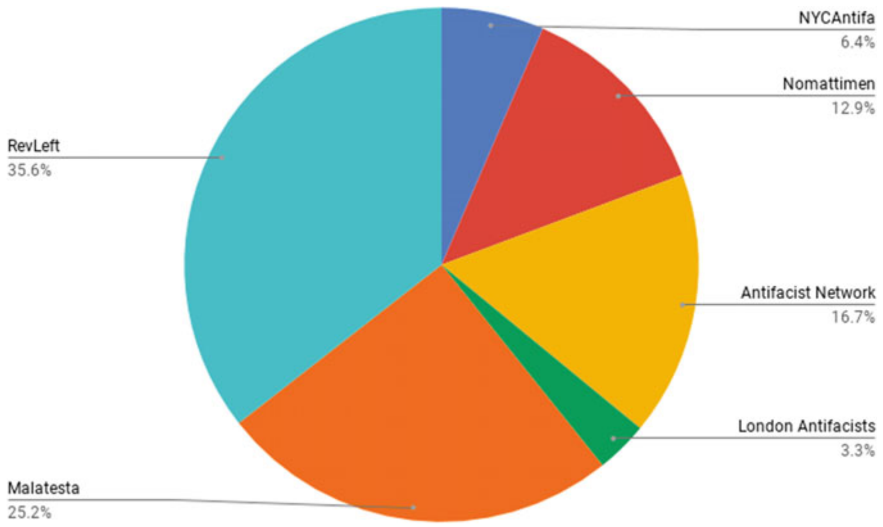


Fig. 13 The breakdown of data for the Antifa dataset

The largest singular source was RevLeft [31], an Antifa forum with a sizeable active population and topics covering the theory, history, and practice of Antifa membership. This site contained approximately 2.8 million postings of which a substantial portion were retrieved. Certain forum categories, such as the "off topic" category, were not added to prevent irrelevant material from being added to the extremist label.

The breakdown of the sources for the Antifa dataset are shown in Fig. 13.

3.4 Sovereign Citizens

Sovereign Citizens are more prolific than other forms of terrorism and a wide variety of content was available. Many of the Sovereign Citizen websites contained both a populated and updated articles section as well as an active forum. For the identified websites, all such relevant material was added to the dataset.

A Free Country [32] is a Sovereign Citizen group with a website that contains both a considerable number of articles as well as an active forum. This content was scraped and analysis was performed to prevent duplicates caused by the articles (which were written by an active forum members) being reposted as forum posts. Hashing-based methods were utilized to ensure retrieved documents were not reproduced in their entirety. This situation appears unique to this site as other sites with both forums and articles did not appear to exhibit the same level of reproduction.

Websites such as Embassy of Heaven [33], FreeMan NZ [34], and Natural Person [35] contain article-style postings with no opportunity for user interaction. These

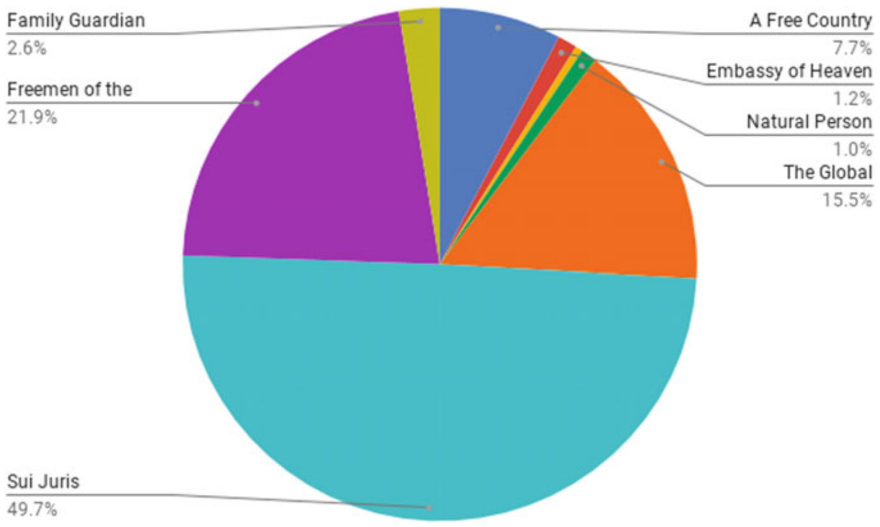


Fig. 14 The breakdown of the Sovereign Citizen dataset

articles cover a variety of instructional material as well as justification and analysis of Sovereign Citizen theory.

A resource titled "The Global Sovereign's Handbook" [36] was identified from a website that tracks terrorist material and individuals. This handbook contained explanations for the origin of the Sovereign Citizen movement as well as detailed instructions on how to resist the government and subvert laws. The handbook totaled approximately 300 pages of content.

Two of the larger Sovereign Citizen forum websites identified were Sui Juris Forum [37] and Freemen on the Land forum [38]. The principal topic of both websites were instructions on how to utilize Sovereign Citizen ideology in interactions with the government and promoting violent resistance to oppose what is viewed as government interference. Specific instructions were given on how to resist legal process, such as traffic stops and search warrants, as well as instructions on how to fight criminal court cases according to Sovereign Citizen theory. A similar but smaller forum, Family Guardian [39] was also included. This forum promoted Sovereign Citizen ideology using an interpretation of Christianity to justify Sovereign Citizen beliefs.

Figure 14 shows the breakdown of the Sovereign Citizen dataset.

3.5 *Benign Material*

For benign material, Johnston and Weiss utilized material obtained from the paste sites that was considered not to be related to Sunni extremist terror in any way; moreover, no such data was identified that was relevant to other forms of extremism.

This dataset was supplemented with datasets of news articles [17] as well as an English copy of the Qu'ran. The intent of such inclusions was to minimize the likelihood that the model would associate the presence of a singular word (e.g. "Syria" or "jihad") with extremism.

For the purposes of this chapter, the benign dataset is the same for all forms of extremism. Given that benign content is relatively poorly-defined compared to extremist material, it would be challenging to develop substantial, unique benign datasets for each form of extremism. A consequence of this strategy is that all curated materials for the benign label are included, regardless of the ideology being tested. During testing, model performance was acceptable without modification of the dataset for each form of extremism, and consequently work into creating distinct benign datasets was not performed.

This chapter utilizes the benign dataset used in Johnston and Weiss [17] but we make substantial additions. For example, the benign dataset was supplemented using a copy of the King James Bible. During experimentation with the model after the publication of the Johnston and Weiss paper, a number of pages quoting or including long passages of the Christian Bible were listed as extremist. This could be the result of the linkage of Islam and Christianity (as both are Abrahamic religions) for Sunni extremist models and more generally the model could be confused due to the violent nature of certain passages. The reasoning behind including the King James Bible matched that of including the Qu'ran: religious content in isolation is not extremist.

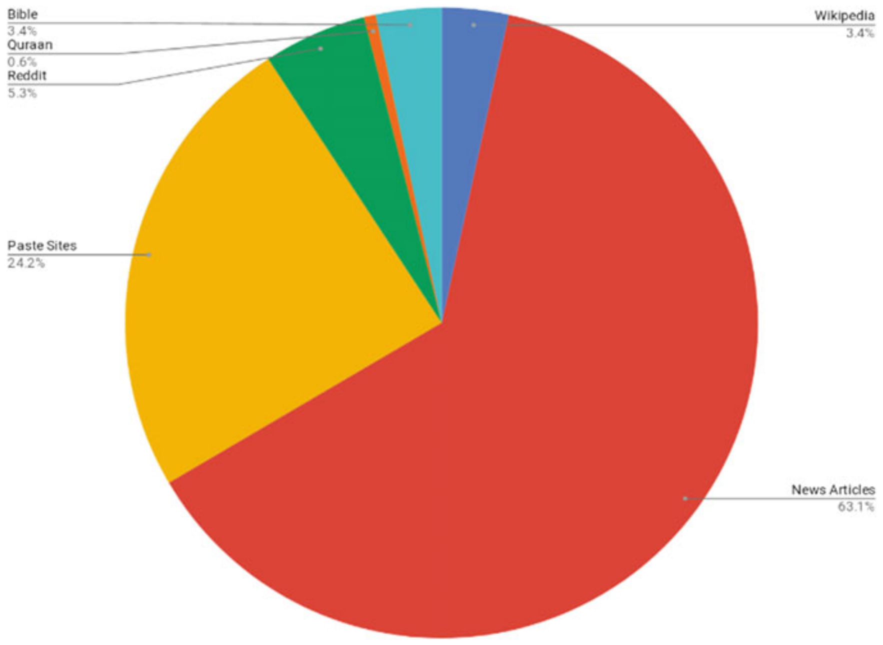


Fig. 15 The breakdown of the benign dataset

Moreover, the benign category was supplemented with an open-source database of Reddit comments [40] and news articles [41, 42] unique from the ones originally added. The intent of adding such data on unspecified topics was to increase the variety and complexity of the benign dataset (Fig. 15) to yield a model more capable of distinguishing extremist content.

4 Building the Models

To demonstrate the necessity of deep learning models, we trained both long-short term memory (LSTM) deep models as well as logistic regression models. To make the data suitable for use, we performed a variety of preprocessing actions dependent on the model. For all models, we removed documents with fewer than 30 words to prevent poor-quality examples from being included.

For LSTM models, we utilize TensorFlow VocabularyProcessor with a maximum document length of 200 words. The input layer takes vectors of size 200, mapping to a dropout layer of 128 neurons with a dropout of 0.5. The dropout layer feeds into a fully-connected layer with softmax activation and the Adam optimizer, with a learning rate of 0.001 and categorical cross-entropy as the loss function. The model was trained for ten epochs with 70% of data being used for training, 10% being used for validation, and 20% for testing.

For logistic regression, Gensim doc2vec was used in the same configuration as Johnston and Weiss [17] except the vector size for doc2vec was configured at 100. The inverse of regularization strength hyperparameter was set to 1.0. Similar to the LSTM model, 70% of data was used for training and 30% of data was used for testing.

5 Results and Analysis

Table 1 demonstrates the results of each model and dataset combination.

Interestingly, the Antifa LSTM model did exceptionally poorly, classifying all data as benign. Considering that logistic regression was able to identify distinctions between each class, the poor performance could be attributed to a dearth of Antifa content or poor hyperparameter choice. However, the same configuration of LSTM performed well with other datasets.

Despite the Antifa's models poor performance, we can see that accuracy is significantly better using the deep model ($p < 0.0001$); however, F1 score is not significantly better ($p > 0.05$). Removing the Antifa results from the calculation, we can see that the LSTM models' F1 score is significantly better than logistic regression ($p < 0.01$). It is reasonable to exclude the poor performance of the Antifa model seeing as the Antifa LSTM degenerated and did not predict any content in the extremist class, a sharp departure from the performance of the other models.

Table 1 Performance by model type and dataset

Model	Dataset	Accuracy	F1 score
LSTM	Sunni	90.69	0.91
	Antifa	83.97	0.77
	Sovereign Citizen	88.65	0.88
	White Supremacist	88.37	0.88
	Average	87.92	0.86
Log. Reg.	Sunni	70.35	0.79
	Antifa	67.46	0.77
	Sovereign Citizen	69.67	0.79
	White Supremacist	65.41	0.76
	Average	68.2225	0.7775

Also worthy of note is that the LSTM model for Sunni extremism here exhibited stronger performance than the deep network described in Johnston and Weiss [17] without the use of a threshold. This better performance can likely be attributed both to the increased dataset as well as the ability of LSTM models to account for information such as word ordering, which is not readily identifiable from the doc2vec output used with the Johnston and Weiss model [17].

As discussed earlier, the benign dataset was kept consistent for all models. Based on the strong performance of the LSTM models, the inclusion of curated benign materials that did not necessarily pertain to the form of extremism (e.g. the King James Bible is unlikely to be confused for Antifa propaganda) did not appear to prevent the model from performing.

For use in real-world systems, such models would likely benefit from the use of a threshold system as described in Johnston and Weiss [17]. These models would likely be of most use when applied to large datasets, where false positives could result in the inefficient allocation of investigative resources. During the research for this chapter, conversations with law enforcement and intelligence professionals were conducted to understand their preferences for the application of such technology. Such conversations indicated that a greater false negative would be preferable to a larger false positive rate within reason. Consequently, imposing a threshold on the LSTM output would likely be of considerable benefit to professionals. Moreover, instead of classification using labels, producing output where the raw scores of the models were utilized could allow law enforcement and intelligence professionals to prioritize materials classified as extremist with greater confidence first, decreasing the chance of encountering false positives until a substantial number of materials were reviewed.

6 Related Work

Some limited work has been done on identifying extremist propaganda using machine learning. Jaki and De Smedt [43] developed models to identify German far-right hate speech on Twitter. Their model had an F1 score of 0.8421 using a single-layer perceptron network with a dataset consisting of 100,000 Twitter posts evenly divided between hate speech and benign content. The hate speech was collected from over a hundred specific far-right accounts. While the performance of this model is strong, the limited number of extremist users could allow a model to overfit to those specific actors. Likewise, the feature vector consisted of character trigrams, word bigrams, and character unigrams. These features would likely not generalize well to multilingual applications and are very vulnerable to obfuscation measures such as the homoglpyh substitution discussed earlier. Futhermore, it is unknown whether these features will generalize well to larger, potentially less structured works. For example, it is possible that word usage frequency will vary when an extremist is not severely limited by character length, as is true for Twitter postings.

Similarly, Ashcroft et al. [44] utilized support vector machines, Adaboost, and Naïve Bayes models to classify Sunni extremist Twitter posts. The dataset was curated using a list of known ISIL-favored hashtags (a Twitter feature allowing users to contribute to a larger discussion on a topic by adding a "#" character before a word) and utilized the presence of such hashtags as well as time-based features. While performance for some models reached 100% accuracy, this work is similarly narrowly-focused on Twitter posts and the features cannot generalize to other web content. Likewise, by curating a dataset utilizing a list of hashtags, the dataset could be biased to a group of extremists within the same organizational structure or sharing a specific interpretation of ISIL ideology.

Gambäck and Sikdar [45] utilized a deep learning approach to classify tweets containing racism, sexism, or a combination. The dataset consisted of approximately 6,700 Twitter posts with a clear majority being neither racist nor sexist in nature. A convolutional neural network using word2vec as a preprocessing step yielded the best results, with an F1 score of 0.78. The poor performance could likely be attributed to the imbalance of the dataset as well as the poorly-defined nature of topics such as sexism and racism. Unlike terrorist ideologies, which are specific in their beliefs, goals, and tactics, sexism and racism can vary depending on cultural context and are likely generationally specific. Although the dataset was divided using both a group of experts as well as a crowdsourcing approach, the labels could have been influenced by the society in which the expert or crowdsourced user resides. Likewise, the size of the dataset is small compared to other studies, especially given the brief nature of tweets. The performance derived from this model would likely not generalize to more unstructured forms of postings.

The work in this chapter is based off the work of Johnston and Weiss [17]. The primary limitations of the work were the narrow focus on Sunni extremism as well as the limited number of models. As mentioned earlier, the models described in the Johnston and Weiss did not include the Christian Bible, so they would miscategorize passages of the Bible as extremist. This work supplements both benign and Sunni extremist datasets, as well as creating models for other forms of terrorism.

7 Future Work

Building models to recognize terrorism is a critical first step in using deep learning to combat terrorism. In order to operationalize these models, they need to be incorporated into a larger system used by law enforcement and intelligence professionals such as forensic analysis software or open-source intelligence tools.

Forensic analysis tools are utilized to review storage media in furtherance of an investigation. Given that large-capacity storage media is becoming increasingly more affordable, an investigator might need to review terabytes of data in order to identify material relevant to an investigation. Utilizing these models as a preliminary review of the material could assist investigators with rapidly triaging material for further analysis. These models could present a novel solution to the problems of forensic analysis of text documents.

An interesting application would be to apply the models to finding intelligence on the darknet. Darknets utilize anonymizing routing softwares such as Tor and I2P to mutually anonymize both the server and client. Consequently, standard search engines are incapable of identifying such material and investigate options are limited as the host of the content cannot be readily identified. Moreover, terror groups have expressed interest in transitioning to anonymizing software; the most notable being the "Cyber Kahilafah" ISIL-affiliated terrorist who regularly published material on how to use Tor to evade law enforcement [46]. These models could be used to empower a crawler that could identify terrorist materials at scale.

Likewise, these models could have application to private industry. Social media websites have an interest in limiting terrorist accounts, but require a scalable solution given the large volume and velocity of posts. These models could be used in unison to detect postings that require additional review before publishing, or to audit accounts nominated for deletion.

A potential extension of this work could include integrating these models with a system to perform author analysis. Malicious actors might take greater steps to remain unattributable across different social networks and platforms, such as using unique usernames or posting under multiple usernames on the same website. An author analysis platform with extremism models enabled could potentially reveal investigate leads that would be challenging to identify without significant manual effort. Likewise, such a system could be used to classify different terrorists by whether they are known to manufacture novel content or reuse content created by others. This information could serve to assist investigators with prioritizing extremists with the knowledge and desire to grow or enable terrorist networks.

8 Conclusion

These models represent significant progress for the cybersecurity and intelligence domains. While great progress has been made in fields such as malware analysis, less effort has been made to identify text content produced by malicious actors. Such text content could be an important source of investigative leads and consequently necessitates a more intelligent, scalable solution than manual investigative methods. While the models in this chapter are limited to identifying extremist propaganda, it stands to reason that such techniques could be used to identify criminal material from other domains.

Likewise, this chapter highlights the relative lack of research within the public data science and cybersecurity community to develop machine learning solutions for law enforcement and intelligence professionals. Terrorism in particular is a rare focus and requires greater attention from the research community to identify solutions that mitigate the spread and capabilities of such actors. As greater numbers of the world population become active online, modern solutions to online investigative problems need to be created in order to handle the large and diverse ecosystem of the modern Internet.

References

1. Our World in Data: Terrorism. https://ourworldindata.org/terrorism (2018). Accessed 15 Jan 2019
2. Abel, D.S.: The racist next door. https://www.browardpalmbeach.com/news/the-racist-next-door-6332282 (1998). Accessed 7 Feb 2019
3. Fisher-Birch, J.: Terror on Tumblr. https://www.counterextremism.com/blog/terror-tumblr (2018). Accessed 6 Feb 2019
4. Alfifi, M., Kaghazgaran, P., Caverlee, J., Morstatter, F.: Measuring the impact of ISIS social media strategy. In: MIS2. Marina Del Ray, California (2018)
5. Department of Homeland Security National Operations Center, Analyst's Desktop Binder, Department of Homeland Security (2011)
6. Roggio, B.: ISIS announces formation of Caliphate, rebrands as 'Islamic state'. https://www.longwarjournal.org/archives/2014/06/isis_announces_formation_of_ca.php (2014). Accessed 13 Jan 2019
7. Operation Pakistan: Letter to Abu Bakr al Baghdadi by 152 leading Islamic scholars. https://operationpakistan.wordpress.com/2014/10/10/letter-to-abu-bakr-al-baghdadi-by-152-leading-islamic-scholars/ (2014). Accessed 6 Feb 2019
8. Oasis Center: the routinization of martyrdom operations. https://www.oasiscenter.eu/en/suicide-bombings-and-martyrdom-in-islam-4 (2017). Accessed 18 Dec 2018
9. Esfandiari, G.: Iran praises 'Martyrdom' of fighter beheaded by Islamic state extremists. https://www.rferl.org/a/iran-praises-martyrdom-fighter-beheaded-islamic-state/28680228.html (2017). Accessed 17 Jan 2019
10. Ransom, J.: White Supremacist pleads guilty to killing black man in New York to start a 'Race War'. https://www.nytimes.com/2019/01/23/nyregion/timothy-caughman-white-supremacist-guilty.html (2019). Accessed 23 Jan 2019

11. Southern Poverty Law Center: KU KLUX KLAN. https://www.splcenter.org/fighting-hate/extremist-files/ideology/ku-klux-klan. Accessed 1 Feb 2019
12. Keller, M.: Reporter shares video of altercation with protester in Charlottesville. https://thehill.com/homenews/media/401480-reporter-shares-video-of-altercation-with-protester-in-charlottesville (2018). Accessed 1 Feb 2019
13. Crowder, S.: Undercover in Antifa: Their Tactics and Media Support Exposed!, Louder with Crowder
14. Federal Bureau of Investigation: Domestic terrorism: the sovereign citizen movement. https://archives.fbi.gov/archives/news/stories/2010/april/sovereigncitizens_041310/domestic-terrorism-the-sovereign-citizen-movement (2010). Accessed 26 January 2019
15. Florida Sherrifs Department: Sovereign Citizen training for law enforcement HS. https://www.youtube.com/watch?v=ALPs_n0WQaY (2015). Accessed 27 Jan 2019
16. University of North Carolina School of Government: A quick guide to sovereign citizens (2013). Accessed 14 Jan 2019
17. Johnston, A.H., Weiss, G.M.: Identifying Sunni extremism using deep learning. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–6 (2017)
18. SITE Intelligence Group. https://siteintelgroup.com/. Accessed 11 Jan 2019
19. Just Paste It: JustPasteIt. https://justpaste.it/
20. Add Post It: AddPostIt. https://addpost.it/
21. Clarion Project: Clarion Project Inc. https://clarionproject.org/. Accessed 17 Jan 2019
22. Stormfront: Stormfront. https://www.stormfront.org
23. Vanguard News Network. https://vnnforum.com/. Accessed 10 Feb 2019
24. The American Freedom Party. http://theamericanfreedomparty.us. Accessed 13 Jan 2019
25. Davidson, T., Warmsley, D., Macy, M. and Weber, I.: Automated hate speech detection and the problem of offensive language. In: Proceedings of the 11th International AAAI Conference on Web and Social Media, Montreal, Canada (2017)
26. NYC Antifa | Culture of resistance against fascism, NYC Antifa. https://nycantifa.wordpress.com/. Accessed 12 Feb 2019
27. Nomattimen: https://nomattimen.wordpress.com/. Accessed 09 Feb 2019
28. Antifacist Network: https://antifascistnetwork.org/. Accessed 01 Feb 2019
29. London Antifacists: https://londonantifascists.wordpress.com/. Accessed 01 Feb 2019
30. Malatesta: https://malatesta32.wordpress.com/. Accessed 25 Jan 2019
31. Revleft: https://www.revleft.space/vb/. Accessed 29 Dec 2018
32. A Free Country: http://afreecountry.com/. Accessed 6 Jan 2019
33. Embassy of Heaven: http://www.embassyofheaven.com/. Accessed 18 Jan 2019
34. Freeman NZ: https://www.freemannz.net/. Accessed 12 Jan 2019
35. Natural Person: http://www.natural-person.ca. Accessed 30 Dec 2018
36. Liberty, J.: Sovereign Citizens Handbook. https://www.trackingterrorism.org/resource/sovereign-citizens-handbook-pdf (2004). Accessed 2 Jan 2019
37. Sui Juris Forum: https://www.suijurisforum.com/. Accessed 5 Feb 2019
38. Freemen on the Land: http://forum.fmotl.com/. Accessed 16 Jan 2019
39. Family Guardian: https://famguardian.org. Accessed 18 Jan 2019
40. Pushshift: Directory Contents. http://files.pushshift.io/reddit/comments/ (2017). Accessed 13 Nov 2018
41. UCI KDD: Reuters-21578 Text Categorization Collection. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html (1999). Accessed 22 Oct 2018
42. Kaggle: All the News. https://www.kaggle.com/snapcrack/all-the-news (2018). Accessed 27 Nov 2018
43. Jaki, S., De Smedt,.T.: Right-wing German hate speech on Twitter: analysis and automatic detection. Hildesheim, Germany (2018)
44. Ashcroft, M., Fisher, A.: Detecting Jihadist Messages on Twitter. In: 2015 European Intelligence and Security Informatics Conference (EISIC). Uppsala, Sweeden (2015)

45. Gambäck, B., Sikdar U.K.: Using convolutional neural networks to classify hate-speech. In: Proceedings of the First Workshop on Abusive Language Online, Vancouver, Canada (2017)
46. Jihadi Hacking Group Cyber Kahilafah Uses Telegram to Inform Pro-ISIS Followers of Private Communication and Impeding Cyber Attack; Highlights Use of Online Information Sharing Platforms. http://cjlab.memri.org/lab-projects/monitoring-jihadi-and-hacktivist-activity/jihadi-hacking-group-cyber-kahilafah-uses-telegram-to-inform-pro-isis-followers-of-private-communication-and-impeding-cyber-attack-highlights-use-of-online-information-sharing-platforms/ (2016). Accessed 10 Jan 2019

Index

A
Advanced driver assistance systems, 117–121, 134, 137, 139
Audio detection, 181
Automated driving, 117–121, 123, 128, 134, 137, 139

B
Baby cry detection, 171, 172, 182, 192
Big-data, 2, 3, 6, 8, 9, 20

C
Convolutional neural networks (CNNs), 24, 33, 35, 56, 61–65, 68–72, 74, 77–85, 87, 91, 93, 96, 100, 110, 118–120, 123–126, 128, 129, 131, 135, 136, 138, 139, 171–174, 176–181, 189–192, 225, 239, 245–249, 251, 260, 262, 197, 206–208, 210, 211, 215, 216, 218–220, 285

D
Deep belief networks, 31, 33, 38, 77, 82, 83, 118, 206
Deep learning, 31–37, 56, 57, 61–64, 69, 73, 74, 77, 79–82, 84, 85, 87, 91–94, 99, 110, 117–120, 138, 139, 145, 152, 160, 162, 164, 171–176, 182, 189, 192, 193, 197, 202–206, 215, 219, 220, 223–227, 229, 231, 232, 234, 238, 239, 245, 247, 249–253, 255–257, 259, 260, 262, 263, 267, 271, 276, 283, 285, 286
Deep neural network, 1, 2, 49, 71, 91, 93, 94, 96, 99, 102, 110, 125, 172, 215, 225
Diagnosis systems, 81, 82, 85
Dimension reduction, 1, 3, 7–10, 12, 13, 18, 24, 25, 27, 157
Domain adaptation, 91–97, 99, 100, 105, 110, 111, 160, 250

E
End-to-end communication, 223, 226, 227, 238
Extremism, 271, 272, 276–278, 281, 282, 284–286

H
Healthcare, 61, 64, 77, 86, 87

I
Industrial applications, 31, 32
Industrial control systems, 197, 198, 200, 202
Injection attacks, 197, 198, 200, 202–204, 209

L
Learning regime, 1, 8, 17
Logistic regression, 171, 172, 174, 182, 186–188, 192, 283
Long Short-Term Memories (LSTM), 139, 182, 192, 260, 267, 283, 284

W. Pedrycz and S.-M. Chen (eds.), *Development and Analysis of Deep Learning Architectures*, Studies in Computational Intelligence 867,
https://doi.org/10.1007/978-3-030-31764-5

M
Machine learning, 5, 31, 36, 37, 73, 78, 79, 91, 94, 96, 101, 122, 154, 164, 173, 175, 192, 202, 205, 223–225, 233, 239, 247, 249, 251, 252, 256, 259, 262, 285, 287

P
Pedestrian detection, 117, 120–125, 128, 130–133, 135–139, 215
Physical layer, 225, 226, 262

R
Regression, 20, 21, 32, 34–36, 50, 84, 91–101, 103, 105, 110, 111, 126, 171, 172, 174, 182, 186–188, 192

S
Semi-supervised learning, 33, 40, 41, 98
Signal detection and classification, 223, 238, 239, 245, 251, 262
Soft Sensors, 31, 32, 36, 40
Speaker embedding, 146–148, 152, 155–159, 164
Speaker recognition, 145, 147, 148
Speaker verification, 148
System identification, 32, 35

T
Terrorism, 267, 269, 271, 272, 277, 280, 286, 287
Text classification, 256

W
Wireless security, 224, 260, 262
Wireless systems, 223–225, 231, 263