2019

Explainable Neural Networks based Anomaly Detection for Cyber-Physical Systems

Kasun Amarasinghe
Virginia Commonwealth University

EXPLAINABLE NEURAL NETWORKS BASED ANOMALY DETECTION FOR CYBER-PHYSICAL SYSTEMS

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

KASUN AMARASINGHE

Bachelor of Science, University of Peradeniya, Sri Lanka, 2011

Director: Milos Manic,

Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

December, 2019

Acknowledgements

First, I would like to thank my advisor and mentor Prof. Milos Manic for his help, support, and guidance. I would like to thank Dr. Preetam Ghosh, Dr. Bridget McInnes, Dr. Supathorn Phongikaroon, Dr. Craig Rieger, and Dr. Ronald Boring for serving in my dissertation committee and for their valuable feedback. Further, I would like to thank Mr. Kevin Kenney for his assistance and valuable comments. I would also like to thank all the collaborators at other institutions and my colleagues for their help and support. Further, I would like to acknowledge the support given to me by the Idaho National Laboratory.

I extend my gratitude to all the great colleagues of the Modern Heuristics Research Group that I was lucky to work with—Dumidu, Jake, Patrick, Daniel, Chathuri, Morgan, and Javier—for their continuous support throughout my Ph.D. journey. While we get to acknowledge people contributed scientifically in publications, we seldom get to appreciate the people who help us with emotional support. To that end, I would like to thank my friends who are in Richmond and Sri Lanka, for being extremely supportive throughout my journey, especially during frustrating and difficult times.

I want to thank my partner, Rushika, for her support, and her sacrifices during my academic journey. I want to thank my sister, Saumya who was also reading for her doctorate, for constantly reminding me why we both were on this journey. Last but by no means the least, although no words will ever suffice, I want to thank my parents, Peter and Malani Amarasinghe, for their endless sacrifices. Without them, none of this would have been possible. I dedicate my doctoral dissertation to my parents, Peter and Malani Amarasinghe.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

EXPLAINABLE NEURAL NETWORKS BASED ANOMALY DETECTION FOR CYBER-PHYSICAL SYSTEMS

By Kasun Amarasinghe

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2019.

Director: Milos Manic,
Professor, Department of Computer Science

Cyber-Physical Systems (CPSs) are the core of modern critical infrastructure (e.g. power-grids) and securing them is of paramount importance. Anomaly detection in data is crucial for CPS security. While Artificial Neural Networks (ANNs) are strong candidates for the task, they are seldom deployed in safety-critical domains due to the perception that ANNs are black-boxes. Therefore, to leverage ANNs in CPSs, cracking open the black box through explanation is essential.

The main objective of this dissertation is developing explainable ANN-based Anomaly Detection Systems for Cyber-Physical Systems (CP-ADS). The main objective was broken down to three sub-objectives: 1) Identifying key-requirements that an explainable CP-ADS should satisfy, 2) Developing supervised ANN-based explainable CP-ADSs, 3) Developing unsupervised ANN-based explainable CP-ADSs.

In achieving those objectives, this dissertation provides the following contributions: 1) a set of key-requirements that an explainable CP-ADS should satisfy, 2) a

methodology for deriving summaries of the knowledge of a trained supervised CP-ADS, 3) a methodology for validating derived summaries, 4) an unsupervised neural network methodology for learning cyber-physical (CP) behavior, 5) a methodology for visually and linguistically explaining the learned CP behavior.

All the methods were implemented on real-world and benchmark datasets. The set of key-requirements presented in the first contribution was used to evaluate the performance of the presented methods. The successes and limitations of the presented methods were identified. Furthermore, steps that can be taken to overcome the limitations were proposed. Therefore, this dissertation takes several necessary steps toward developing explainable ANN-based CP-ADS and serves as a framework that can be expanded to develop trustworthy ANN-based CP-ADSs.

CHAPTER 1

INTRODUCTION

The desire to create machines that can think dates back centuries [1]. In the present day, we have made great strides toward creating intelligent machines with a thriving research area dubbed artificial intelligence (AI). AI has enabled smart assistants that understand speech, software that translates languages, and cars that drive themselves (see Figure 1).

At the core of modern AI-based systems are algorithms that enable machines to learn from experience and use that knowledge to perform new tasks without explicit instructions. In the early days of AI, the focus was on solving problems that can be described in a list of mathematical rules. It was soon recognized that the real challenge was in solving problems that cannot be formalized into a set of rules but are easy for humans to solve intuitively. The solution was building machines that learn from prior experience—learn from data. Therefore, AI became data-driven and spawned the field of machine learning. The recent rapid progress in AI is largely owed to a class

(a) (b) (c) (d)

Fig. 1. Ubiquity of AI in the modern world. (a) 'Siri' [2], (b) 'Google Assistant' [3] , (c) Self-driving cars [4], (d) Google translator [5]

of machine learning algorithms named Artificial Neural Networks (ANNs). ANNs are biologically inspired and use their multi-layered structure to learn high-level abstract representations of data. Further, they are capable of learning higher-order features with minimal human intervention [1]. Therefore, ANNs are capable of learning very complex patterns that exist in data. As a result, ANNs have transformed fields such as image recognition, speech recognition, and natural language processing [6].

Despite the recent improvements in AI, there is a lack of trust in AI among humans [7]. As a result, there is a reluctance to fully adopt AI in safety-critical systems such as medical diagnoses and cybersecurity [8]. In safety-critical applications, often automation needs to collaborate with humans-in-the-loop, and trust in deployed AI is extremely important. Therefore, garnering the benefits of modern AI in safety-critical applications depends on how well the humans and AI systems co-exist. Therefore, building 'trustworthy AI' is necessary to ensure a harmonious relationship between AI systems and human operators in these systems. Accordingly, as ANNs are the core of modern AI, developing trustworthy ANN-based systems is necessary.

The principal reason behind the lack of trust is ANNs' inability to explain their decisions and being perceived as black-boxes [7–9]. The black-box nature prevents users from understanding the reasons behind a decision made by the ANN and what the ANN has learned from data. As a result, users have no indications of whether the ANN makes predictions based on strong evidence or artifacts in data [10], [11]. Therefore, an essential step to build trust in AI/ANN systems is to crack open the said black box. Revealing what the ANN/AI has learned and the reasons behind ANN/AI outputs is essential in achieving the goal of trustworthy AI/ANNs. This desired quality of explaining the knowledge and decision-making process of ANNs is named 'explainability' of ANNs [12].

Developing explainable AI systems has become a highly popular research topic.

Explainable AI research can take two main approaches: 1) developing novel machine learning algorithms that learn explainable features, 2) developing methodologies that explain existing machine learning algorithms. Defense Advanced Research Projects Academy (DARPA) spawned an array of projects named Explainable Artificial Intelligence (XAI). This initiative mainly takes the first approach to explainable AI research [12]. This involves designing and implementing novel machine learning algorithms that combine the learning capability of complex ANNs and the explainability of models such as Decision Trees. This dissertation takes the second approach to explainable AI research.

The most popular type of XAI research is generating visualizations of saliency maps (heat-maps) for image classification problems [13–15]. However, humans are more inclined to justify things verbally [16]. Hence, textual explanations would resonate more with humans than visualizations. There hasn't been much work in deriving linguistic explanations for ANNs or ANN predictions. Hendricks et al. presented a methodology for generating textual explanations for image classifications using a combination of ANN algorithms [17]. However, the method couldn't guarantee that the ANN used the features described in the explanation for classification [18]. Therefore, to the best of our knowledge, there is a gap in existing research for generating textual explanations of ANNs.

The theoretical focus of this dissertation is developing explainable ANNs. However, this dissertation argues that the notion of explainability and requirements of explainability are highly domain and problem-dependent. Therefore, this dissertation focuses on a specific application domain. The application domain of choice is anomalous behavior detection in Cyber-Physical Systems (CPSs). Therefore, the focus of this work is developing methodologies for explaining ANN-based anomalous behavior detection in CPSs.

1.1 Motivations

This section presents the motivations for the theoretical focus and the application domain focus in the dissertation.

Motivations for theoretical focus: Creating trustworthy AI systems is one of the grand challenges of AI and would impact the way we adopt AI in the future. Black-box models are the biggest factors standing in the way of developing trustworthy AI and reaping the benefits of AI in fields with direct human impact. Therefore, opening the black-boxes through explanation is essential. As ANNs are the core of modern AI, developing explainable ANNs serves the grand objective of creating trustworthy AI systems.

Motivations for application domain focus:

CPSs integrate computational and physical resources for the optimized management of physical resources. CPSs are at the core of modern critical infrastructure ranging from transportation systems, power grids to space exploration systems [19–21]. Their significance makes CPSs prime targets of adversaries with malicious intent and any successful intrusion could result in catastrophic consequences. Not only malicious intrusions but also benign faults (e.g. equipment failure) could lead to similar results. Therefore, monitoring for any anomalous behavior—malicious or benign—is crucial for ensuring security, reliability, and resiliency of CPSs. Data-driven techniques are well-suited for such dynamic environments and the vast amounts of data produced by CPSs make it possible. ANNs' superior pattern recognition capability makes them ideal candidates for the task. However, deployed methods being explainable is a necessity for the domain. Therefore developing explainable and ANN algorithms for CPS anomaly detection serves the grand objective of securing our infrastructure.

1.2 Objectives of the Dissertation

The main objective of this dissertation: Development of explainable ANN based Cyber-Physical anomaly detection systems (CP-ADSs).

The main objective is broken down into three **sub-objectives**:

1. Identification of key desired features of an explainable CP-ADS
2. Development of supervised ANN-based explainable CP-ADSs
3. Development of unsupervised ANN-based explainable CP-ADSs

The first sub-objective of this dissertation is to identify a set of key requirements that an explainable CP-ADS should satisfy. Once these features/requirements are identified, they can be used to evaluate "explainability". In this dissertation, we argue that the desired features in an explainable system can't be generalized and should be defined by taking into account end-users and the unique properties of the problem domain. Despite increasing interest in explainability, there is no consensus on what explainability in machine learning is, how to evaluate it, or what well-formed evaluation strategies are [22], [23]. Therefore, this is a crucial first step in achieving the main objective of the dissertation.

CP-ADSs often use a combination of supervised and unsupervised algorithms [24], [25]. Therefore, this dissertation considers both cases. Accordingly, the second sub-objective is developing supervised ANN-based explainable CP-ADS. The focus of the work is narrowed to classification based supervised CP-ADS. The role of the generated explanations is to help the end-users understand what the CP-ADS has learned about each anomaly type in its training phase and answer the following question:

> "*What system behavior is considered by the CP-ADS to detect a certain anomaly type?*"

The third sub-objective is developing and testing unsupervised ANN-based explainable CP-ADS. CPSs generate massive amounts of data that are mostly unlabeled. These unlabeled data streams can be leveraged with unsupervised learning and the CP-ADS is trained to identify clusters of behavioral patterns. The role of the explanations is to help the user understand the different behavioral patterns in the CPS and answer the following question:

> "*What system behavior is prominent in a certain CPS behavioral pattern?*"

1.3 Contributions of the Dissertation

In the process of achieving the said objectives, five contributions are presented in this dissertation.

First, the requirements for an explainable CP-ADS are presented. The set of requirements is discussed based on the end-user requirements and unique properties of the problem domain.

Second, a methodology for summarizing the knowledge of a supervised ANN trained as a CP-ADS is presented. The presented methodology enables deriving a linguistic summary of what the neural network has learned about each anomaly type in its learning phase. Further, several metrics are presented that can be used to assess the quality of the derived summaries.

Third, a methodology for quantitatively validating the derived explanations is presented. The presented methodology creates artificially perturbed input instances—adversarial examples—to validate the derived explanations. The introduced perturbations based on the generated explanations and the CP-ADS' response to the artificially perturbed input instances are used to validate the explanations.

Fourth, a novel unsupervised ANN algorithm for identifying behavioral patterns

of a CPS is presented. The presented methodology enables using unlabeled data to learn the different states that the CPS goes through. Furthermore, the methodology enables the generation of visualizations to explain the behavioral patterns.

Fifth, a methodology for linguistically explaining the identified behavioral patterns using the unsupervised neural network is presented. The derived explanations describe the different behavioral states to the user and help the users understand why the behavioral patterns are clustered. The visualizations generated through the unsupervised ANN are combined with the linguistic explanations.

1.4 Organization of the Dissertation

Chapter 2 provides an overview of CPSs and ANNs. First, the chapter introduces deep neural networks, outlines the general learning algorithm and how neural networks can be used for anomaly detection in CPS. Next, the chapter briefly overviews the prior work in explaining neural networks and identifies the gap in research. Then, the chapter introduces CPSs, their general architecture and the problem of anomaly detection in CPS.

Chapter 3 presents the first contribution of the dissertation. This chapter presents a discussion about explainability and its requirements in the context of CPS anomaly detection. The chapter identifies the unique properties of CPS anomaly detection and discusses what it means to be 'explainable' in that context. The chapter concludes with recognizing a set of key requirements an explainable CP-ADS should satisfy.

Chapter 4 elaborates on the second and third contributions of this dissertation, i.e. the methodology for explaining what a supervised neural network has learned and the methodology for validating the explanations. First, the overall methodology for deriving linguistic explanations from the trained neural network is presented. Then, the methodology for evaluating the explanations is discussed. Next, the results

obtained by testing the methodologies on several datasets are presented. Finally, the presented methodology is evaluated against the requirements identified in Chapter 3 and, the successes, limitations of the method and possible next steps to overcome the next steps are presented.

Chapter 5 elaborates on the fourth and fifth contributions of the dissertation, i.e. the novel unsupervised neural networks algorithm for learning behavioral patterns of a Cyber-Physical System and the methodology for explaining the identified behavioral patterns. First, the basis of the novel deep unsupervised neural network, Self-Organizing Maps (SOMs) is introduced and the learning algorithm of SOM is presented. Then, the chapter elaborates on the novel Deep Self-Organizing Maps (DSOMs) algorithm and its usage for identifying behavioral patterns of a CPS. Next, the methodology for explaining the knowledge gained by the DSOM is presented. The chapter is concluded by presenting results obtained by implementing the methodologies on several real-world datasets.

Chapter 6 presents the overall conclusions of the dissertation and discusses future directions of the work. Furthermore, this chapter presents a brief discussion about the future of explainable intelligence and how related machine learning research areas could coincide with the goals of explainability from the author's point of view.

CHAPTER 2

BACKGROUND

This chapter discusses the background information required to follow the work presented in this dissertation. In addition, this chapter presents a succinct overview of literature with respect to explainability of Deep Neural Networks.

2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs), more recently rebranded as Deep Neural Networks (DNNs), are biologically inspired algorithms that can learn high-level abstract representations in data. ANNs have the ability to learn from data with a hierarchy of concepts that builds from simple concepts to generate larger concepts. This capability enables the machine to learn complex concepts or patterns in data starting from raw data. Conventional machine learning algorithms had limited capability in dealing with data in their raw form. Pattern recognition tasks involved feature engineering to transform the raw data into a suitable representation for the machine learning algorithms. Conversely, in DNNs, the ability to learn high-level abstract representations in data with multiple layers enables learning very complex patterns in data with minimal human intervention. As a result, DNNs have revolutionized AI research and has become the most important area of research in building intelligent machines.

ANNs are layered architectures with input, hidden and output layers. While traditional ANNs were commonly used with a single hidden layer, DNNs of today are capable of learning with a large number of hidden layers, due to the advancements in

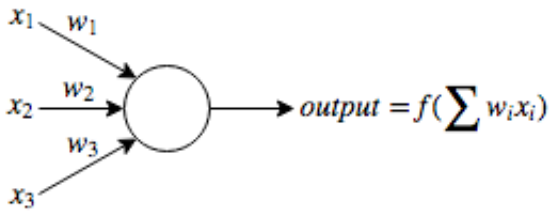


Fig. 2. An artificial neuron

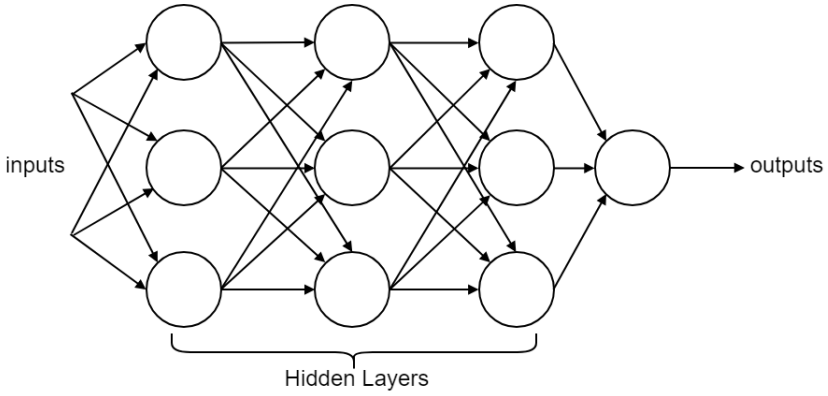


Fig. 3. Neurons arranged in layers—input, hidden and output

neural networks research, advancement in computing power and data storage capabilities. Therefore, essentially, DNNs are ANNs with a large number of hidden layers. Today, the terms ANN and DNN are used synonymously.

The building block of a DNN is an 'artificial neuron and ANNs/DNNs are loosely based on biological neural networks. Similar to a biological neuron, an artificial neuron receives a set of inputs and produces an output based on inputs (Figure 2). The weighted sum of the inputs is generally transformed with a nonlinear function to generate the input. This function is called the activation function or the transfer function. The output is called the activation of the neuron. These neurons are arranged in layers to make up a neural network. As mentioned, these neurons are arranged in layers, namely input, hidden, and output (See Figure 3).

DNNs are trained using a training dataset and their learning is entirely data-driven. The knowledge of a DNN is stored in its weights. The neural networks learn by adjusting their weights to optimize a cost function. The most popular learning algorithms for neural networks is error back-propagation [26]. The DNNs learn iteratively and at each step, the error between the produced output of the DNN and the target output is calculated and the error is propagated through the layers to change the weights accordingly to minimize the error. This optimization process is usually performed by a gradient-based approach such as gradient descent or ADAM [27]. Therefore, the calculations are done in a forward pass, and the learning of the weights is done in a backward pass. For a detailed description about the learning in DNNs, readers are referred to [1].

2.2 Explaining Neural Networks

This section presents the concept of explainability in DNNs and briefly overviews the recent efforts in literature toward explaining DNNs.

It has been widely agreed upon that the main reason behind the lack of trust in modern AI systems is the black-box nature of the models and the lack of explainability stemming from that [8], [7]. David Gunning of DARPA stressed the importance of developing explainable AI systems and DARPA has been working on developing a new suite of explainable AI (XAI) algorithms [12]. In the XAI project, the focus is to develop new algorithms that have the learning capability of a DNN and the explainability of models like decision trees.

It is important to distinguish that there are two types of explanations for AI systems: 1) explaining an individual prediction and 2) explaining the overall model. In a similar vein, there are two definitions of trust in the context of AI: 1) trusting an individual prediction, i.e. whether a user trusts a specific output from the model so

that an action can be based on it, and 2) trusting a model, i.e. whether the user trusts the model to act in the expected way to deploy it. The XAI project aims at developing AI models that can answer the following questions system as one that answers the following questions: 1) Why did it do that? 2) Why didnt it do something else? 3) When does it succeed? 4) When does it fail? 5) When can it be trusted? and 6) how can an error be corrected? [12]. In terms of the two types of trust/explanations, the first two questions fall under explaining an individual prediction, and the rest falls under explaining an overall model.

Since the modern AI systems are primarily driven by DNNs, creating explainable DNNs has been a topic that has received much attention in recent years. Despite the interest, explaining DNNs effectively remains to be an open research area. In the literature, the words explainability and interpretability are used interchangeably. However, in this dissertation, we remain consistent with the definitions given in Chapter 1. The notion of explainability of machine learning models is not a monolithic concept. Explainability can be viewed from two angles, 1) model transparency and 2) model functionality [18], [16].

Transparency of the model refers to understanding what the network has learned and the reasons behind the concepts it has learned. Transparency can be viewed in three parameters: 1) decomposability, 2) simulatability, and 3) algorithmic transparency [18]. Decomposability is whether there is an intuitive explanation for the model parameters. Algorithmic transparency relates to the ability to explain the inner workings of the learning algorithm. Simulatability refers to the ability of a human using the input data together with the model to reproduce every calculation thats necessary to make the prediction, allowing a human to understand the changes in the model parameters during the training process. Given the complexity of DNNs, achieving these three components is not a trivial task. Further, it is assumed that

the simulatability is very low in DNNs and hence most of the research is focused on improving decomposability and algorithmic transparency [18].

Model functionality explanations can be used to explain predictions by the model. This facet of interpretable DNNs is also called post hoc explanation generation [16]. Post-hoc explanation generation entails understanding a pre-trained model, i.e. the trained model is available and methods attempt to gain a functional understanding of the trained model [28]. Post-hoc explanations can be generated in three different ways. The first method is to provide textual justifications of the DNN predictions. This involves providing a semantically meaningful description of the models output and the reasons behind the output. Therefore, it requires a combination of models. The second method is to provide justifications through different visualizations of parameters. In the third method, local explanations are used to gain insight into the models behavior. For instance, in DNNs the gradient of the output with respect to the inputs can be used to identify the local changes that are influenced by the input vector [29]. This type of explanations are the most explored in the literature.

One of the first attempts to explain DNNs through visualization was made by Erhan et. al. [13]. The authors developed a methodology where the function of each hidden neuron could be visualized using a method called activation maximization. The method facilitated a way to visually analyze the hidden units of the DNN by finding the input records that maximize the activation of the hidden unit in question. The authors implemented this method on Deep Belief Networks and Autoencoders to compare and contrast the features the hidden units of the networks were learning.

After Erhans study, there were several other methods proposed to generate heat maps especially for image classification, to understand the most important parts of an image for classification. Sensitivity analysis was the first method to be proposed for identifying the inputs that the output was most sensitive to [29]. In this method, the

sensitivity of a pixel is calculated by calculating the partial derivatives of the output with respect to the pixel. This results in a local sensitivity score where it indicates how much a small change in the pixel would affect the output. This method was used to analyze Convolutional Neural Networks in [14]. This heat map is not generating an explanation as to why the output was generated but gives an indication of the inputs that the output is sensitive to.

Zeiler and Fergus proposed another heat map visualization technique for CNNs using deconvolution operations [15] called deconvolutional heat maps [30]. In this method, deconvolution operations are carried out in a backward pass to map the activations from the network back to the pixel space of the image. This results in a heat map where pixel values indicate their relevance to the output activations. This method is limited to CNNs with ReLU units and max-pooling.

Layer-wise relevance propagation is a method that was proposed by Bach et al [10] for decomposing a classification decision into pixel-wise relevance scores. These relevance scores is a measurement of their contribution to the output classification score. This methodology can be used for any DNN with monotonous activation functions [11]. This method can be used to generate pixel-wise relevance for each individual classification decision.

In non-visual explanation generation methodologies, one of the most leading methods is Local Interpretable Model-Agnostic Explanations (LIME) proposed by Ribeiro et al. [8]. LIME provides a method to explain any classifier and gives the user a binary vector with the same length as the input vector. A one indicates that the corresponding input feature was used in the classification by the model and vice versa. LIME generated explanations for an individual classification decision by generating samples in the local vicinity of the input record in question and approximating an interpretable discriminator (E.g. linear classifier) that is faithful to those points.

That interpretable discriminator is not globally faithful. LIME deals with explaining individual predictions and is slow because of the example points generation.

Textual explanations tend to resonate more with humans because we often justify things verbally. In one of the first attempts to generate textual explanations of image classification, Hendricks et. al proposed a method to classify the image and provide an explanation of the image [17]. The methodology was inspired by the automatic caption generation methodologies [31]. The method uses a combination of CNNs and RNNs to perform the classification and the explanation. The network jointly optimizes the classification and selecting the optimal explanation for the for the image. Even though the model has shown impressive results, the method does not guarantee that the classification is done using the features that are described in the explanation. The explanation is the textual description that best matches with the identified features. Therefore, the method cannot be used for gaining insight into algorithms that will be deployed in safety-critical systems.

The above review presented the most notable contributions in the field of explaining DNNs. In the interest of brevity, only the notable contributions that help gauge the state-of-the-art in the field were presented. A more comprehensive review can be found in [16] and [18]. It can be seen that most of the work available in the literature is focused on explaining individual classification decisions and generating visual explanations. The only textual explanation generation study for DNNs does not guarantee that the explanation will reflect the reasons behind the classification. In contrast, the goal of this dissertation is to develop methodologies to provide textual explanations of the overall model prior to deployment.

2.3 Cyber-Physical Systems

Cyber-Physical Systems (CPSs) are systems with highly integrated physical and computing resources that can interact with humans through a range of methods [20], [32]. The operations of the physical resources are monitored, controlled, and managed by computing resources such as embedded computers with a communication network playing the role of integrator. The control strategies usually contain feedback loops where the physical processes and embedded controllers affect each others function [33]. CPSs enable physical entities to collaborate and communicate with each other to optimize the control of each component based on knowledge of each other.

The development in physical resources and the ability to seamlessly connect with the rapid growth in networking technologies have created opportunities for CPSs in every stratum of modern society. Smart grids, traffic control, medical devices, smart buildings are just a few of examples of systems with tightly integrated physical resources and computing [34], [35]. Further, The highly interconnected nature of CPSs enable technological advances in a multitude of areas, including personalized healthcare; emergency response, manufacturing, and energy management [32], [20], [36]. Almost all CPS are geographically distributed and as a result, employ distributed control and management strategies, contain a plethora of sensors for sensing the CPS state, actuators for controlling physical processes, communication devices and control units.

CPSs encompass a multitude of existing technologies such as embedded systems, distributed control systems, and communication networking systems. Therefore, CPSs contain the attributes of its components. It is recognized that the CPSs have attributes that mainly include, distributed control, heterogeneity, real-time operation (timeliness), security, reliability, scalability, and autonomy [37], [38]. Therefore,

technologies that are developed to monitor and control CPSs should satisfy the unique requirements of CPSs.

2.3.1 General Architecture of Cyber-Physical Systems

CPS architectures are very application specific and highly complex [38]. However, given the general components that are present in all CPSs, a general architecture can be presented. As mentioned above, the CPS consists of physical resources, computing resources, and communication networks. The operations of a CPS can be broken down to four operational layers; 1) Physical layer, 2) Sensors and actuators layer, 3) Network layer, and 4) Control layer [39], [38]. In addition to the four layers, an information layer can be thought of as encapsulating all four layers as information flows through all the layers [38]. Figure 4 shows the general architecture of a CPS with its five layers and their connections to one another.

The *physical layer* consists of the resources in the physical world. These resources are the ones being monitored and controlled by the CPS. Examples for physical layer components would be power generators, transmission lines for a power system, buildings and their components for modern buildings in smart grids and connected buses and trains in an intelligent transportation system of the future.

The *sensor-and-actuator layer* sits directly on top of the physical layer. Sensors fulfill the role of measuring the state the physical system. These measurements are sent through the information layer to the above layers. The actuators are responsible for translating the control signals from the above layers to the physical system. For a power grid, the sensor information can be currents, voltages and phase angles of buses while the actuator control signals can be open/close breaker.

The *networking layer* is responsible for connecting the sensor actuator layer to the control layer. This layer contains the communication network with communication

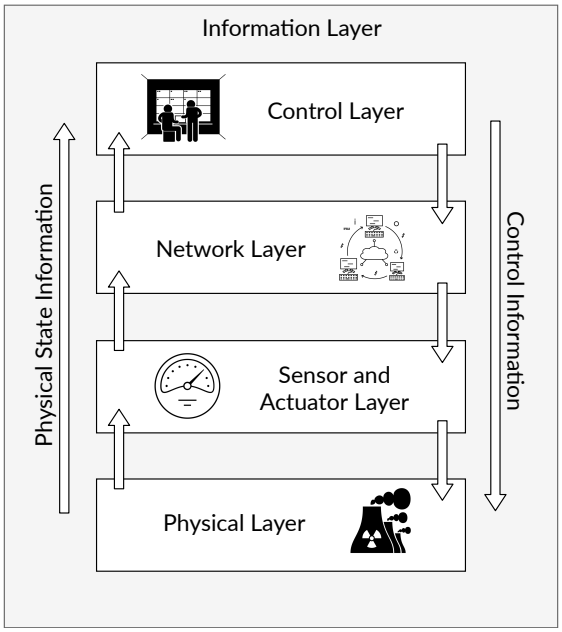


Fig. 4. General architecture of a Cyber-Physical System with four operational layers. The information layer encapsulates the four operational layers. Adapted from [38]

devices and protocols. This layer handles networking protocols such as TCP-IP, DNP3, and MODBUS. The network layer is responsible for delivering the sensor information to the control layer and delivering control signals to the actuators.

The *control layer* makes the control decisions about controlling physical resources through distributed control practices and supervisory control. Since most CPSs are geographically distributed, distributed controllers create local feedback control loops for their local physical resources with the use of remote terminal units, programmable logic controllers and intelligent electronic devices. The supervisory control layer creates a system-wide global feedback control loops by aggregating data from multiple points in the network. In addition to the feedback control loop, the supervisory control layer incorporates human-in-the-loop control strategies where the global CPS measurement data are monitored by human operators.

The *information layer* is an abstract layer that encapsulates all the other layers in the CPS. Sensor measurements and the control decision are the main types of information that traverses through the CPS and uninterrupted and accurate information flow is paramount to ensure optimal function of the CPS.

2.3.2 Anomaly Detection in Cyber-Physical Systems

Anomaly detection is the process of detecting data points that do not conform to the expected behavior of data and is also named outlier detection, novelty detection, deviation detection and exception mining [40]. An anomaly is defined as a set of observations that are inconsistent with the other observations in data [41]. Anomalous observations in data can be caused by a multitude of reasons such as system faults, human error, changes in system behavior and fraudulent activity.

In the context of CPSs, and anomalous behavior in data can be a result of a random disturbance such as an equipment failure, or it could be the result of a malicious attack from an adversary to the communication network or the physical resources. As CPS are ubiquitous in most industrial systems, including critical infrastructure such as the nations power grid, any disruption in their performance could lead to catastrophic and cascading damages. Therefore, regardless of the cause, it is of utmost importance that all measures are taken to prevent such disruptions.

In order to minimize random equipment failure, CPSs maintain preventive maintenance procedures and fault-tolerant algorithms. Threats from malicious adversaries can be categorized into physical attacks and cyber attacks. As the names suggest, physical attacks refer to direct attacks on the physical resources and cyber attacks refer to adversaries gaining access to the communication network(s) of the CPS. For both these attacks/intrusions types, rigorous prevention methodologies are implemented by CPSs. For example, to prevent physical attacks, mechanisms such as

access control and surveillance are in place. In order to prevent cyber attacks, CPSs employ prevention techniques based on cryptography.

While intrusion prevention techniques are necessary to secure CPSs, they alone are not sufficient. For example, adversaries who gain legitimate privileges to the system will not be caught using preventive measures alone. Therefore, no matter how strong the intrusion prevention systems are, detection systems are necessary to catch any intrusion or failure that slips through the prevention mechanisms. It has been identified that CPSs require ability to detect and report anomalous behavior [42]. Therefore, anomaly detection is an crucial area of research in CPSs.

Anomaly Detection System design for CPSs need to take into account the unique properties of CPSs such as their distributed nature. For instance, large-scale CPSs will require an ensemble of methods detecting anomalies due to their heterogeneity and distributed nature. In resource-constrained components, simple and fast detection anomaly detection methodologies will have to be used. The more complex and highly accurate anomaly detection methodologies can be deployed in components with fewer resource constraints, such as controllers in the control layer. Therefore, DNN based anomaly detection methodologies can serve the complex detection process while simple techniques are deployed in individual components.

Existing CPS anomaly detection techniques can be broadly categorized into two categories: 1) knowledge-based and 2) behavior-based. In knowledge-based approaches, the anomaly detection happens by pattern matching of anomalies [43]. This approach is also referred to as misuse detection and supervised detection [44, 45]. The main advantage of this method is the low false positive rates. However, knowledge-based methods rely on having a predefined set of misbehavior signatures to perform pattern matching. With evolving adversaries, this is an almost impossible task, and thus knowledge-based methods are not suitable for anomaly detection in CPSs.

Behavior-based anomaly detection techniques look for out of the ordinary behavioral patterns in run-time features [45]. Typically the ordinary behavior is learned using data-driven techniques. Machine learning based anomaly detection techniques fall under this category. Supervised learning methods, unsupervised learning methods or semi-supervised learning methods could be used in this type of anomaly detection. Several machine learning algorithms such as genetic programming [46], Bayesian classifiers [47] and neural networks [48], unsupervised clustering algorithms [49], [50] have been proposed for anomaly detection. In addition to the machine-learning based approaches, graph-based approaches [51–53] , statistical approaches [54–56], and signal and image processing techniques [57], [58] have been applied in CPS anomaly detection.

The main advantage of behavior-based approaches is that they are not looking to match specific misbehaviors. This eliminates the need for maintaining an exhaustive list of attack signatures. Further, behavior-based methods have the capability of detecting previously unseen intrusions or failures. However, the main disadvantage of this type is the relatively high false positive rate. Therefore, behavior-based techniques should be developed to minimize the false positive rate while maximizing the detection rate.

CHAPTER 3

EXPLAINABILITY OF CYBER-PHYSICAL ANOMALY DETECTION SYSTEMS

As mentioned, explaining black-box models has received a considerable amount of attention in the recent years. However, overwhelming majority of the literature focus on developing new methodologies for generating explanations using the developer's intuitions without much emphasis on the intended end-users [59], [60], [22]. As a result, there is no consensus on what explainability in machine learning is, how to evaluate it, what well-formed evaluation strategies are, or the desiderata [22], [23].

In this dissertation, we argue evaluation strategies and desired features differ across application domains and user groups, and hence, it is very difficult to design a set of generalized requirements for explainable machine learning. Therefore, in this chapter, we attempt to identify a set of necessary requirements of an 'explainable' CP-ADS. It should be noted that defining an exhaustive set of requirements is a process where the actual system users need to weigh in. Therefore, we are not claiming these conditions to be sufficient.

This chapter attempts to identify the requirements of an explainable CP-ADS with respect to the type of machine learning algorithm, the point in time where the explanation is generated, the target end-user, and the medium of explanation. In other words, this discussion will be around the following questions:

1. *What* is being explained?

2. *When* is the explanation generated?

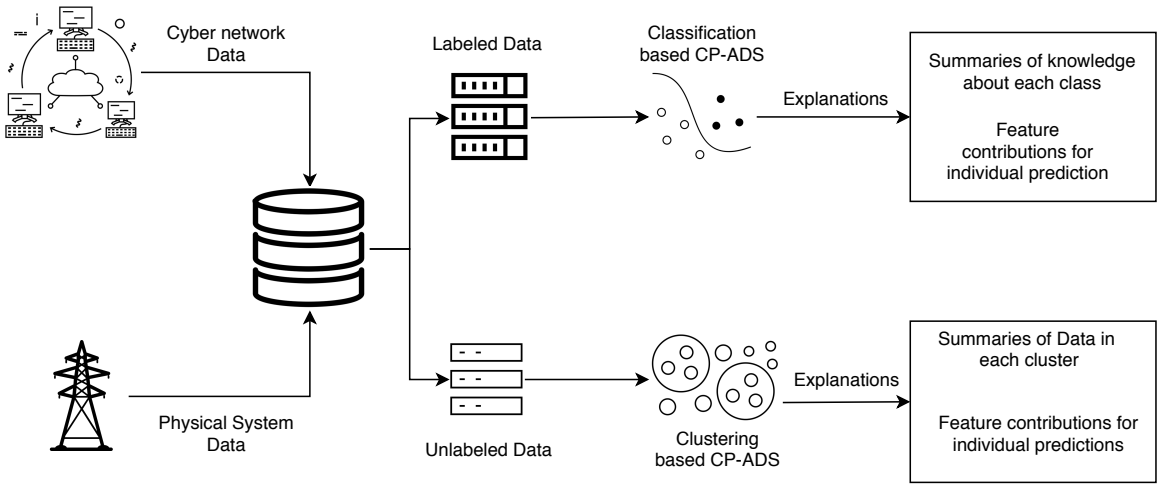


Fig. 5. The overall picture of developing explainable CP-ADS. The main steps and components

3. *To whom* is it being explained?

4. *How* is the explanation communicated?

3.1 Explanations and the Type of Machine Learning Algorithm: *What?*

In this section, how the type of learning algorithm (unsupervised vs supervised) used in CP-ADS affects the explanation process is discussed.

In the context of CP-ADS, usually, a combination of supervised and unsupervised machine learning techniques are used in tandem [24], [25], [61]. Therefore, it is important to discuss the notion of explainability for both supervised and unsupervised methods. In this work, for simplicity, we consider supervised methods to be classifiers and unsupervised methods to be clustering methods. These terms are used interchangeably in this dissertation.

Supervised CP-ADS are trained to classify between normal behavior classes and anomalous behavior classes and results in low false positive rates compared to unsupervised methods [62], [63]. Labeled training data are required for all the intrusion

types (concepts) it is trained to detect. Therefore, the model is learning a specific set of patterns (concept). Therefore, the explanations have to be related to the concepts that the CP-ADS is being trained to identify. The explanations should enable end-users to understand what the CP-ADS has learned about each intrusion type it is trained to detect.

Unsupervised CP-ADSs learn from unlabeled data. Obtaining labeled data to train supervised methods is costly [64]. Further, it is virtually impossible to anticipate every anomaly type (malicious or benign) that could happen to a system. Unsupervised CP-ADSs learn behavioral patterns using data and then flag any previously unseen behavior as an anomaly [65], [25]. In this scenario, there are no clearly defined concepts. Therefore, the goal of explainability is to understand the data [66]. The explanations should enable users to understand the different behavioral patterns the model has learned.

3.2 Explanations and the Timeline of CP-ADS: *When?*

In this section, we discuss the timeline of a CP-ADS and the different types of explanations that should be generated at different points in that timeline.

Figure 6 shows the general process of a CP-ADS. The complete process has four main steps: 1) training the CP-ADS using historical data, 2) testing and validating the model, 3) deploying the model, and 4) monitoring the CPS through live predictions.

As mentioned, the main objective of generating explanations is making human operators trust the CP-ADS. It is important to note that, this context, there are two type of trust: 1) trust in the overall model, and 2) trust in the individual predictions [8]. Accordingly, two types of explanations need to be generated: 1) explanations of the overall model, and 2) explanations of the individual prediction. These two types are directly related to the timeline of a CP-ADS process. With respect to the

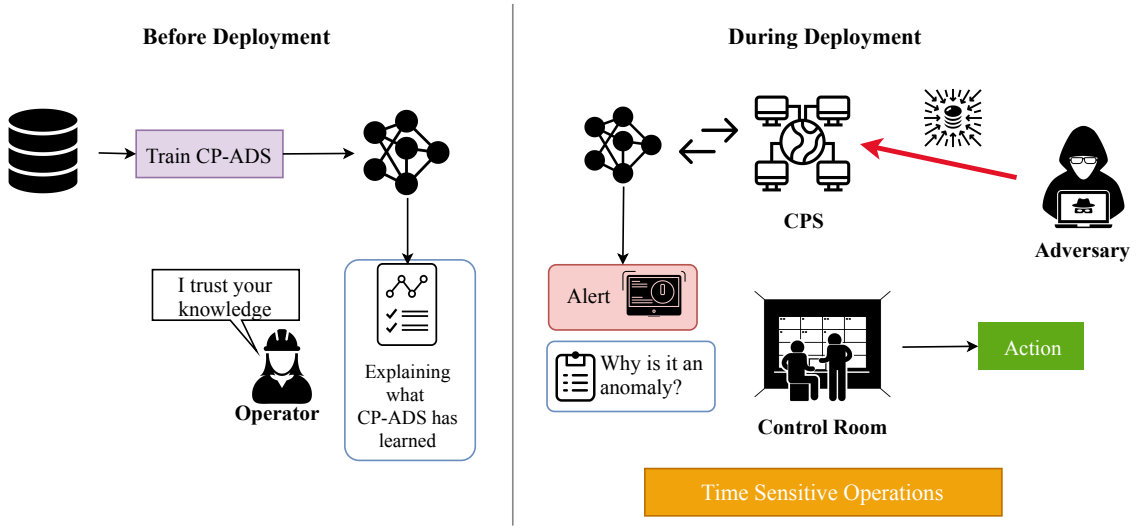


Fig. 6. The complete process of an explainable CP-ADS. The process has two phases on either side of deployment: 1) before and 2) during. Explanations should be generated in both phases and the goals of explanation are different in the two phases. It should be noted that the explanation and action process during deployment is very time sensitive.

four-step process, the overall model explanations are generated in the second step and individual prediction explanations will be a part of the fourth step.

CPSs are safety-critical systems and human operators should be able to trust an the overall CP-ADS prior to deploying it in the system. Therefore, overall model explanations are extremely crucial in this context. Once deployed, when CP-ADS detects and anomaly, evasive mitigation steps need to be taken [67]. Since safety is paramount, this is very time sensitive and does not permit a thorough examination of a prediction explanation. Therefore, out of the two explanation types, we argue that overall model explanations are more important in the context of CP-ADS. It should be noted that this is not the case for all application domains. For instance, consider the case where an AI model is used to determine the eligibility of an applicant for a loan. If the loan is rejected, there should be an explanation about that individual

prediction. In such applications, individual predictions are more important than overall model predictions.

When the CP-ADS algorithm is highly non-linear such as a neural network, generating overall model explanations is very difficult [8], and the state-of-the-art methods mainly focus on explaining individual predictions (Chapter 2). It is important to discuss the two types of explanations in terms of the two types of CP-ADS.

Supervised CP-ADS: In a supervised CP-ADS, the concepts are well defined. Therefore, the overall model explanations should summarize what the CP-ADS has learned about the intrusion types it is trained to detect. For instance, the cyber-network behavior that generally influence the CP-ADS to detect a Denial-of-Service (DoS) attack can be an explanation for the DoS concept. For individual prediction explanation, the input features that contributed most to that prediction can be used as the explanation (feature attribution). As mentioned in Chapter 2, there are several methodologies that can provide this E.g. LRP [28], LIME [8].

Unsupervised CP-ADS: As mentioned, the explanation goal of an unsupervised CP-ADS is to understand data. Therefore, the overall model explanations should summarize what the CP-ADS has learned about the clusters it identified. For instance, this could be a summary of the input feature behavior patters in the cluster. Individual prediction explanations, input features that contributed most to assigning the instance to a particular cluster can be used as the explanation.

It should be noted that due to the contextual importance and the gap in research, the focus of this dissertation is generating overall model explanations.

3.3 Explanations and the End-user: *To whom?*

In this section, we discuss the explanation requirements from the end-user's point of view.

The goal of generating explanations in the context of this dissertation is building trust in the human operators. Thus, the intended end-user is the human operator that monitors and controls the CPS. Therefore, the explanations should be understood by the human operators.

It is worth noting that the explaining and understanding are two different concepts [23]. The process of explaining is dependent on the model that is being explained, and understanding completely depends on the person receiving the explanation. That means, the expertise level and the cognitive process of the user is directly tied into how the explanation is being perceived, and the specific user group (operator vs engineer vs manager) should be taken into account when explanations are being generated.

First and foremost, the explanations should be tied to the real system components. For instance, in a ANN model, the components that has a tie to the real system are the inputs and the outputs of the system. Therefore, the explanation has to be presented to the user in terms of the inputs to the ANN (E.g. sensor readings from the physical system, network-traffic attributes). If the inputs to the ANN are high level or abstract features, there should be a methodology to trace it back to the real-system components .

A few recent papers called for using social science models in conjunction with machine learning research to generate human-friendly explanations [68] [59], [69]. Miller argued that for explanations to simulate human cognitive process, explainable AI community should learn from human sciences. In his extensive study, Miller presented two major findings from explanations in human sciences that can be applied to CP-ADS: 1) explanations should be contrastive, and 2) explanations are selective and should focus on one or two possible causes, not all causes [68]. These findings lead to the notion that explanations need to be *interpretable*.

The terms *interpretation/interpretability* and *explanation/explanability* are used interchangeably in a majority of research. However, it is important to distinguish between the two.

According to the merriam-webster dictionary, to interpret is to "explain the meaning of" or "present in understandable terms" [1]. Since the goal is to distinguish between the terms explain and interpret, the latter definition is more applicable in this context. A formal definition was presented by Montavon et. al [28]:

> *"An interpretation is the mapping of an abstract concept (e.g., a predicted class) into a domain that the human can make sense of"*

Therefore, in this work, interpretability is defined as a desired quality of a generated explanation, i.e. answering the question "is the generated explanation presented in understandable terms to the human?".

While Miller argued that explanations need not have all the causes but should focus on a one or two causes [59], Gilpin et al. argued that there is a tradeoff between interpretability and completeness [69]. When explanations become more complete, the interpretability of the explanations decrease. Therefore, Gilpin et al. suggest that there should be a mechanism for the user to choose the balance between the two. In this context, we agree with [69] and argue that the user should be able to choose the complexity of the explanations generated. This leads to addressing the need of addressing different expertise levels of different user types. For instance, a cyber-security expert might be able to grasp a highly complex and long explanation with all causes but a high-level manager might just need one or two causes.

Therefore, the ultimate human-friendly explanations for CP-ADS need to be 1) mapped to the real-system, 2) contrastive, and 3) should enable the users to balance

[1] https://www.merriam-webster.com/dictionary/interpret

the completeness with the interpretability of the explanations.

3.4 Explanations medium: *How?*

Another important aspect that is closely related with the human-friendliness is the medium which the explanation is presented. In existing methods for explaining individual predictions, visualization methods are used in terms of saliency maps and heat maps [28].

However, when generating overall model explanations, we argue that textual explanations make it more interpretable since humans often justify decisions verbally [16]. More complete an explanation gets, more information the explanation will contain. However, a visualization limits the number of different dimensions that can be presented to the user. Therefore, a textual explanation can contain more information.

However, visualizations are quickly grasped by humans over text [70]. Therefore, for individual prediction explanations, due to the time sensitivity we discussed in the previous section, a simplified explanation presented in terms of a visualization can help build trust while meeting the urgency requirement.

In an ideal system, the user should be able to choose the medium which the explanation is delivered. In this work, we argue that for overall model explanations, textual explanations are more suitable and for individual prediction explanations, visual explanations are more suitable.

3.5 Desired Features of an Explainable Cyber-Physical Anomaly Detection System

In this section we summarize the above discussions to come up with a succinct set of necessary requirements that a CP-ADS should satisfy to be explainable.

1. The CP-ADS should be able to generate two types of explanations. 1) expla-

nations of the overall model, 2) explanations of individual predictions

   (a) Supervised CP-ADS: Summarize what the model has learned about each concept (class) and explain each classification decision through feature attribution

   (b) Unsupervised CP-ADS: Summarize the input feature behavior in each cluster and explain the features that contribute most to individual cluster assignment

2. All explanations should be human-friendly

   (a) Explanations should be presented in terms of the real system behavior

   (b) Explanations should be contrastive

   (c) User should be able to choose the balance between interpretability and completeness

3. All explanations should be delivered as a visualization or textual explanation

   (a) Overall model explanations should be textual due to high dimensionality of information

   (b) Individual prediction explanations should be visualizations and simple to facilitate immediate understanding

   (c) In an ideal framework, the user should be able to pick the explanation medium they want

4. An evaluation strategy should accompany the explanation methodologies

   (a) A quantitative evaluation strategy to evaluate the correctness of the explanations

(b) A qualitative evaluation strategy with end-user groups to evaluate the effectiveness in the real-world

3.6 Conclusions

This chapter attempted to identify a set of key requirements that an explainable CP-ADS should satisfy. The discussion was based on explainable machine learning research and social science research. The unique properties of the problem domain were considered and four key requirements were identified in terms of timeline, human-friendliness, explanation medium and the evaluation requirements of the generated explanations. It has to be noted that these requirements are not exhaustive and we don't claim them to be sufficient. However, we argue them to be necessary conditions and a much-needed starting point. To identify the requirements of specific systems, it is crucial to have end-users' input as they are the main stake-holder of the explanations. These key areas identified in this chapter should be expanded with further inter-disciplinary research.

CHAPTER 4

EXPLAINING SUPERVISED NEURAL NETWORKS TRAINED FOR ANOMALY DETECTION

In this dissertation, supervised CP-ADS systems are considered to be classifiers, i.e. in this chapter, ANN based classification is considered the CP-ADS methodology. ANN classification has been shown to be a potent form of anomaly detection in a range of domains [48], [71], [72]. If labeled data are available for normal and anomalous behavior, the superior pattern recognition capabilities of supervised neural networks can detect anomalies with high detection rates and low false positive rates [62], [63]. An ANN based anomaly detection system (NN-ADS) is trained to distinguish one or more specific anomaly types from one or more normal scenarios. As mentioned, the concepts the NN-ADS learns are well defined with labeled data.

This chapter presents a methodology for explaining what the NN-ADS has learned in its training phase. As mentioned in Chapter 3, the focus of the work is generating overall model explanations. The presented explanation methodology summarizes what the NN-ADS has learned about each anomaly type (concept) that it was trained to detect. The presented methodology derives linguistic explanations about each anomaly type.

The methodology presented in this chapter assumes an already trained NN-ADS. For simplicity, the methodology assumes a Feed-Forward Neural Network (FFNN) based NN-ADS. However, the methodology extends to other neural network architectures as well. As mentioned, in CPSs, anomaly detection entails detecting cyber-physical (CP) anomalous behavior that could indicate a malicious intrusion or a be-

nign fault. However, for simplicity, the presented methodology is discussed in terms of cyber anomaly detection (intrusion detection) in this chapter. The methodology applies to physical anomaly detection and cyber-physical anomaly detection.

In addition to the explanation methodology, this chapter presents a methodology for validating the derived explanations. The validation methodology is based on systematically introducing perturbations to data. The presented explanation and explanation validation methodologies were implemented on several datasets and the results are discussed in this chapter.

Further, the presented CP-ADS methodology is evaluated against the set of requirements that Chapter 3 presented to identify the successes and limitations of the presented method. Steps are proposed to overcome the identified limitations. Furthermore, we present a discussion on how to extend and scale the method.

This chapter first, presents the methodology for summarizing what the NN-ADS has learned. Second, the presented explanation validation methodology is elaborated. Next, the experimentation details are presented. Then, a discussion of the explanation methodology's capabilities, its limitations, and further methods to extend and scale is presented. Finally, the chapter is concluded with conclusions and possible next steps.

4.1 Methodology for Explaining What the Neural Network has Learned

As mentioned, the supervised NN-ADS is a classifier and is trained to detect a predefined set of intrusions. Each intrusion type is a class/concept of the NN-ADS. In this work, the goal of the explanation process is to summarize what the NN-ADS has learned about each concept. The output of the methodology is a linguistic description for each concept (concept description). Each concept description contains a set of system behavior that is considered to be relevant by the NN-ADS to detect

that concept. In other words, the concept description reflects which input features and values indicate the presence of that intrusion type according to the trained NN-ADS.

The final concept descriptions are derived from IF-THEN type linguistic summaries generated using a test dataset. The IF-THEN linguistic summaries indicate the association between feature values and the relevance of that feature to the presence of a certain intrusion. First, a brief background on Linguistic Summarization and Layer-wise relevance propagation is presented. Next, the methodology of deriving IF-THEN linguistic summaries is presented. Then, the method of creating the final concept description is presented.

4.1.1 Linguistic Summarization

This section briefly introduces Linguistic Summarization (LS). If the reader is familiar with the basics of LS techniques, this section can be skipped.

Linguistic Summarization (LS) was introduced by Yager [73]. LS enables extraction of useful patterns in large multi-dimensional datasets and presenting them in human friendly linguistic descriptions. LS has been shown to be extremely useful in a range of applications [74], [75], [76].

Consider a dataset D with M instances, where each instance is a collection of input features $X = \{x_d\}$, where d is the d^{th} feature of the dataset. LS techniques can be used to extract patterns that exist for $X \in D$. Linguistic summaries are derived using Type-1 fuzzy sets introduced by Zadeh [77]. First, all the features of the dataset are fuzzified into a preset number of fuzzy set by mapping input values x to a *degree of belonging* to each fuzzy set (membership degree), which is denoted as $\mu_s(x)$. The number, shape, and limits of fuzzy sets are defined to satisfy the user and domain requirements. Higher the number of fuzzy sets used, more descriptive the summaries are. However, if the number of fuzzy sets is increased beyond a certain

point, understandability of the summaries is reduced [78].

There are two types of linguistic summaries: 1) IF-THEN type and 2) Yager type. In this work, we use IF-THEN type linguistic summaries. An IF-THEN type linguistic summary takes the following form:

$$\text{IF } d_a \text{ IS } S_1 \text{ AND } d_b \text{ IS } S_2 \text{ THEN } d_c \text{ IS } S_3 \tag{4.1}$$

where $a \neq b \neq c$ and d_a, d_b and d_c are dimensions of the dataset. d_a and d_b are the antecedents with fuzzy sets S_1 and S_2, respectively, and d_c is the consequent with fuzzy set S_3.

The $\mu_s(x)$ values are used for assessing the quality of the linguistic summaries. There are several quality measurements proposed in literature that can be used to rank the derived summaries. The quality measures used in this paper are introduced in the next section.

4.1.2 Feature attribution for individual predictions using Layer-wise Relevance Propagation

This section briefly introduces the layer-wise relevance propagation (LRP) methodology used to quantify the relevance of each input feature to each classification decision made by the NN-ADS.

In this work, linguistic summaries are derived based on the input features and their local relevance to classification decisions. The local relevance score is a quantitative measure of the contribution each input feature made, to the detection of each class/concept. The input feature local relevance is calculated for individual classification decisions using the LRP method.

LRP was introduced by Bach et al. as an approach for understanding pixel-wise contributions to image classification [10]. LRP assumes that the classification

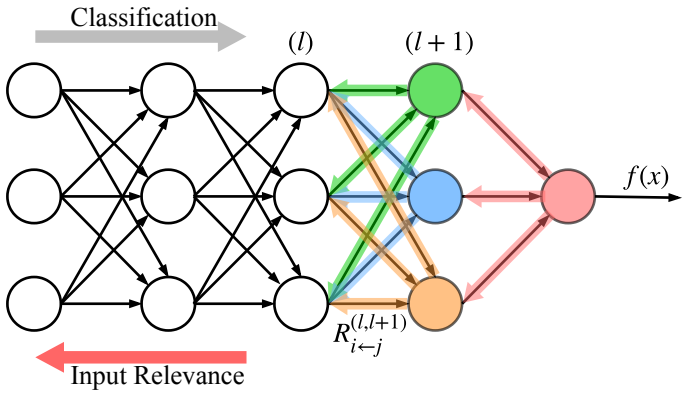


Fig. 7. Layer-wise relevance propagation method. The relevance score for each neuron is backpropagated through the layers

algorithm can be decomposed into several layers of computation. Inputs layer is considered the first layer and the classification layer is considered the last layer. Each neuron of each layer has a relevance score ($R_d^{(l)}$) where d is the neuron (dimension) and the l is the layer. In the input layer, the neuron becomes an input feature. The relevance scores are propagated backward through the layers, i.e. the goal is to calculate relevance scores for layer l when relevance scores for layer $(l+1)$ are available. Figure 7 depicts the process of propagating relevance scores through the layers.

As mentioned, the multi-layered architecture of the NN is leveraged to propagate the relevance scores in a single backward pass, by expressing lower level relevance scores as a function of upper-level relevance scores. Relevance scores are back-propagated in "messages", $R_{i\leftarrow j}^{(l,l+1)}$ (from neuron j in $l+1$ to neuron i in l). The relevance propagation has to satisfy the following relevance conservation properties.

$$f(x) = \cdots = \sum_{d\in l+1} R_d^{(l+1)} = \sum_{d\in l} R_d^{(l)} = \cdots = \sum_{d} R_d^{(l)} \tag{4.2}$$

$$\sum_i R_{i\leftarrow j}^{(l,l+1)} = R_j^{(l,l+1)} \tag{4.3}$$

$f(x)$ is the output and $R_d^{(1)}$ is the relevance score of the d$^{\text{th}}$ dimension of the input layer. The relevance score of the i$^{\text{th}}$ neuron in layer l can be expressed as:

$$R_i^{(l,l+1)} = \sum_j R_{i\leftarrow j}^{(l,l+1)} \tag{4.4}$$

This relevance scores are distributed based on the ratio of pre-activations as follows:

$$R_{i\leftarrow j}^{(l,l+1)} = \left(\frac{a_i^{(l)} w_{ij}^{(l,l+1)}}{\Sigma_i a_i^{(l)} w_{ij}^{(l,l+1)} + b_j^{(l+1)}} \right) \cdot R_j^{(l+1)} \tag{4.5}$$

One drawback of the above rule is a small activation in the j $^{\text{th}}$ neuron makes the relevance scores unboundedly large. The $\alpha\beta$ method [10] is used to counter that drawback:

$$R_{i\leftarrow j}^{(l,l+1)} = \left(\alpha \frac{a_i w_{ij}^{+}}{\Sigma_i a_i w_{ij}^{+} + b_j^{+}} + \beta \frac{a_i w_{ij}^{-}}{\Sigma_i a_i w_{ij}^{-} + b_j^{-}} \right) \cdot R_j^{(l+1)} \tag{4.6}$$

Where $a_i w_{ij}^{+}$ and b_j^{+} are the positive portions of the activations and the negative portion is indicated by a superscripted "-". It has to be noted that the superscripted layer notations have been stripped off in the above equation to help the readability , but notations i and j are considered to be indices associated with layers l and $l+1$ respectively.

Therefore, with the above propagation rule, the relevance scores can be propagated to the first/input layer, i.e. the relevance score of each input feature d; $R_d^{(l)}$ can be obtained. A positive $R_d^{(l)}$ indicates that dimension d supports the existence of the detected concept and vice versa. This results in a quantitative measurement of the contribution of each individual input feature to the classification decision made by

the NN-ADS.

4.1.3 Deriving IF-THEN Type Associative Linguistic Summaries

LS techniques are used for deriving IF-THEN type linguistic summaries. The summaries are derived with respect to one concept. A single linguistic description expresses the association between input feature values and their influence to detecting the selected concept. The antecedents of the description represent feature values and the consequent represents the level of influence. An example linguistic description is given below.

$$\text{IF } f_1 \text{ IS } low \text{ AND } f_2 \text{ IS } low \text{ THEN } inf \text{ IS } high \tag{4.7}$$

where f_1 and f_2 are input features and the low is a Type-1 fuzzy set that indicates feature values. The fuzzy set configuration is entirely at the discretion of the users. The consequent inf is the influence of the antecedent behavior on detecting the intrusion type. Therefore, the linguistic descriptions can be viewed as a list of sub-regions in the feature space that are prioritized by the NN-ADS to detect each intrusion.

Several quality measures can be calculated for each IF-THEN type linguistic summary. Quality measures indicate the 'goodness' of each summary. These measures are used to filter the 'good' summaries for creating the final concept descriptions. As mentioned in before, the quality measures are calculated using membership degree values of the fuzzy sets. Two main quality measures are considered in this work: 1) degree of truth, 2) degree of coverage.

Degree of truth (d_t) is a measure of correctness. d_t indicates how true the summary is given the test dataset. As with any machine learning task, biases in the test data can affect this metric and will affect the quality of the explanations. Antecedent membership degrees are calculated w.r.t the fuzzified input feature values. Conse-

quent membership degrees are calculated w.r.t to fuzzified influence scores. A high d_t would imply that the summary is correctly identifying the specific feature behavior as influential in detecting the intrusion type. d_t is calculated as follows:

$$d_t = \frac{\sum_{m=1}^{M_c} min(\mu_a^1(v_{m,a}^1), \ldots, \mu_a^n(v_{m,a}^n), \mu_c(r_{m,c}))}{\sum_{m=1}^{M_c} max(\mu_a^1(v_{m,a}^1), \ldots, \mu_a^n(v_{m,a}^n))} \tag{4.8}$$

$$\text{where } r_{m,c} = \sum_{d=1}^{n} I_d \tag{4.9}$$

where, $\mu_a^i(v_{m,a}^i)$, are degrees of membership of the m$^{\text{th}}$ test instance's input feature values to their respective antecedent fuzzy sets, $\mu_c^i(r_{m,c})$ is the degree of membership of the mean local influence scores of the antecedent input features for the m$^{\text{th}}$ test prediction, and M_c is the number of data records classified as the concept in question.

Correctness alone is not sufficient for identifying high-quality summaries. A measurement of generality is necessary. Degree of coverage d_c is used as the generalization measurement. d_c indicates how much of the dataset is correctly represented by the summary. A low d_c can indicate outlier linguistic summaries. d_c is calculated as a non-linear mapping of fraction of data that satisfies the summary. A high d_c would imply good generalization. d_c is calculated as follows:

$$d_c = f_c\left(\frac{\sum_{m=1}^{M_c} t_m}{M_c}\right) \tag{4.10}$$

where:

$$t_m = \begin{cases} 1, & \text{if } min(\mu_a^1(v_{m,a}^1), \ldots, \mu_a^n(v_{m,a}^n), \mu_c(r_{m,c})) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

In this work, the sigmoid function is used as the f_c. This maps the ratio of data points that satisfy the LD to a value between 0 and 1. The shape of the function can be fine-tuned to match the user requirements.

These summaries are derived using a test dataset. Existing methodologies for explaining individual predictions (e.g. heat-mapping and saliency mapping techniques mentioned in Chapter 2) can be used to quantify the influence levels of each feature for each test instance. Therefore, as the first step, the test dataset is used to query the trained NN-IDS and a heat-mapping methodology is used to quantify the relevance of each input feature to each prediction. Multiple test datasets can be used to improve generalizability. The relevance scores derived from heat-mapping methods can take positive or negative values. A positive score indicates a feature supporting the classification, and vice-versa. Before descriptions are generated, the relevance scored are normalized to an 'influence score'. This influence quantity is referred as the local influence score in this dissertation. The normalization can be done in several different ways [79]. In this work, the positive relevance scores are scaled between 0 and 1. Therefore, only the features that supporting the classification are considered. How the negative relevance scores can be used is discussed later in the Chapter.

$$I_d = \begin{cases} \frac{R_d - min(R)}{max(R) - min(R)}; \text{ if } R_d > 0 \\ 0 \qquad\quad ; \ otherwise \end{cases} \tag{4.12}$$

Where, I_d is the local influence score of the d^{th} input feature, R_d is the relevance score from the heat-mapping method, and R is the set of relevance scores for all dimension for a prediction.

Once the local influence scores for all features are calculated for all the test predictions, the IF-THEN type linguistic summaries are derived. The derivation algorithm is presented in Table 1. First, all the input features and their influence scores are fuzzified and mapped to a membership degree. Then, for all possible combinations of fuzzy sets, the linguistic summaries are derived and quality metrics

are calculated. This is done with a brute force approach. Finally, the linguistic summaries are filtered based on the preset quality thresholds. These thresholds are set by the user and can be changed to match user and domain requirements.

4.1.4 Deriving Concept Descriptions

Once the high quality linguistic summaries are extracted for an intrusion type, the concept descriptions are created. The user has the flexibility to adjust the thresholds for d_t and d_c to fit the domain and user requirements. The input feature behavior contained in the high-quality summaries are used to generate the final concept descriptions. The concept descriptions formats explanations into a more comprehensible format.

For example, in a neural network trained to detect Denial-of-Service (DoS) attacks, an example concept description for DoS can be expressed as follows:

> "**High values** for *communication_speed*, *number_of_packets_per_host*, and *number_of_sources* are considered as evidence of a **DoS attack** by the NN-ADS"

where, high *communication_speed*, *number_of_packets_per_host*, *number_of_sources* are feature behavior that indicate the occurrence of a DoS attack to the NN-ADS. The concept descriptions can be adjusted semantically and syntactically to match different user requirements. The feature behavior summarized in the concept descriptions translate to the system behavior that indicate a specific type of intrusion to the NN-ADS.

Table 1. Algorithm for deriving linguistic summaries

Inputs:

Test data: $X := \{X_d\}$

Trained NN-IDS: ANN

Feature FS: $F_a := \{A_i : i = 1, \ldots n\}$

Influence FS: $F_c := \{C_j : j = 1, \ldots m\}$

Quality thresholds: (T, C)

Outputs:

List of high-quality linguistic summaries

```
1:  getSummaries (X, ANN, F_a, F_c, T, C)
2:      list ← {(a, c) : a ∈ F_a, F_c ∈ C} % cartesian product
3:      ŷ ← ANN(X) % test data predictions
4:      {R_d} ← LRP(ŷ) % local relevance
5:      description_list ← empty − list
6:      for d ∈ X % for each feature in X
7:          for all (a, c) in list
8:              S ← IF d IS a THEN inf IS c
9:               // Fuzzification of crisp values
10:             μ_a ← Fuzzify(X_d, a)
11:             μ_c ← Fuzzify(X_c, c
12:             // Quality measures for the LD
13:             truth ← Equation 4.8
14:             cov ← Equation 4.10
15:             If truth >= T  &  cov >= C then
16:                 description_list.append(S)
17:             end if
18:         end for
19:     end for
20: end function
```

4.2 Adversarial Approach for Validating the Explanations

Adversarial machine learning is used to exploit weaknesses of machine learning systems by conducting 'adversarial attacks'. Adversarial attacks use modified input instances called adversarial examples. Adversarial examples are input records with intentionally perturbed input features [80]. Adversarial examples help test the robustness/stability of a trained machine learning algorithm such as a neural network [80], [81]. Typically, the perturbations for adversarial examples are introduced using gradient based optimization methodologies [80], [81] and the idea is to capture a small perturbation in the input that 'tricks' the classifier. These methods have been mostly demonstrated in image classification algorithms and it has been shown that a single perturbed pixel could trick a NN [82]. The success of an adversarial attack indicates that the NN's decision function is actually influenced by the pixel(s) in making the decision.

In this work, we use the principle behind adversarial attacks for validating the derived linguistic explanations. As mentioned, if a perturbation to an input (e.g. pixel in an image) causes the NN to change the output, it is safe to assume that the NN considers that input to be relevant to making the decision. The derived concept descriptions outline the input features and their behavior relevant to each intrusion type. Therefore, these relevant input features can be used to create the adversarial examples. If the identified relevant behavior are correct, the NN-ADS decision should be affected by the perturbed examples.

The adversarial example generation algorithm is presented in Table 2. To validate the summaries derived for a certain concept (intrusion type), the instances classified by the NN-ADS as 'normal' are selected and perturbed As the first step, the input records classified as "normal" are extracted. It is worth noting that the perturbations

Table 2. Algorithm for generating the adversarial examples

Require:

$\hat{y}$, X, $map(d, S_d)$

1: **function adversarialExamples** $(\hat{y}, X, map)$
2: $X_{norm} \leftarrow X[\hat{y} == 0]$ % Data predicted as normal
3: $x \leftarrow rand(X_{norm})$ % randomly sampling records
4: **for** d, s in map **do**
5: $v \leftarrow p$ s.t. $\mu_s(p) = 1$
6: $x[d] = v$ % Perturbations
7: **end for**
8: **return** x
9: **end function**

are applied to instances that are classified as 'normal' by the NN-ADS, not necessarily the actual normal instances. Then, extracted instances are perturbed by introducing the feature behavior deemed to be relevant by the derived concept description. If the identified relevant behavior is correct, the NN-ADS should flip the classification label of these perturbed examples from normal to the intrusion type. That is empirical evidence that the NN-ADS actually considers these behavior as relevant in detecting intrusions of that type.

4.3 Experiments

In this section, the experiments that were conducted to implement the presented explanation and validation methodologies are discussed. First, the fuzzy system configuration used for explanation generation is presented. This section is organized with respect to each dataset. For each dataset, different case studies are presented. Each

experiment is organized into sections describing the data set, classification results, derived explanations, and adversarial validation results.

4.3.1 Fuzzy System used for Explanation

In NN-ADS explanation process, two types of crisp values are fuzzyfied: 1) input feature values, 2) input feature influence scores. In this work, all input features were fuzzyfied into three fuzzy sets—low, medium, and high. In an ideal implementation, the number and shape of fuzzy sets should be individually tailored to each input feature in each dataset. For simplicity, we used the same fuzzy set configuration for all the input features in all datasets. Determining the optimal fuzzy set configuration is discussed later in this chapter.

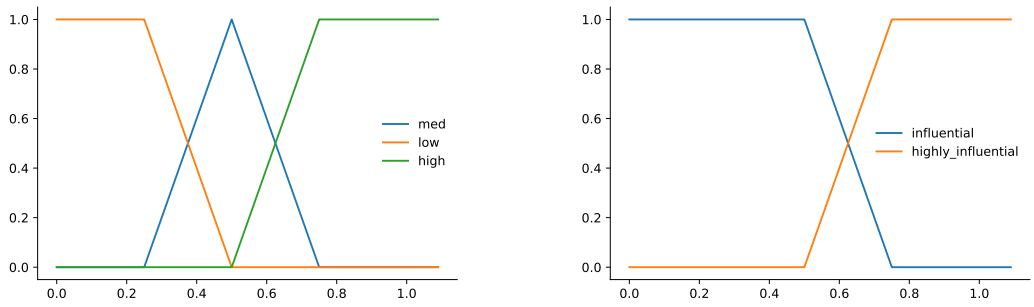
The low, medium and high was set in terms of the quantiles of input feature distribution. The fuzzy set configuration is given in Figure 8(a).

Influence scores were fuzzyfied using two fuzzy sets–influential and highly influential. Ideally, for different datasets, the fuzzy set configuration should be tweaked. But for simplicity, the same fuzzy sets were used for all features and datasets. The fuzzy set configuration is given in Figure 8(b).

4.3.2 NSL-KDD Experiment

The NSL-KDD dataset is arguably the most popular intrusion detection dataset used literature and is an improved version of the KDD Cup 99 dataset [83]. The data were captured during the DARPA IDS evaluation program 1998 and was used in the Third International Knowledge Discovery and Data Mining Tools Competition. Tavallaee et al. created the NSL-KDD dataset by making improvements such as removing redundant records [83].

Each data record in the dataset is a TCP connection and they fall under five



(a) (b)

Fig. 8. Type-1 Fuzzy sets used for explaining the supervised CP-ADS (a) Fuzzy sets used to fuzzify features values, (b) Fuzzy sets used to fuzzify local influence

classes: normal, denial-of-service (DoS), probing, user-to-root (U2R), and root-to-local (R2L). The data distribution in the train and test datasets can be seen in Fig 9. It can be observed that the U2R and R2L classes are extremely underrepresented in the dataset. Therefore, the data records from R2L and U2R classes are not considered in this study.

Each TCP connection record consists of 41 input features. These features consist of basic features acquired from the TCP connection, traffic features acquired from a window of two seconds and content featured acquired from the application layer data. In terms of data types, the 41 features contain 7 categorical features (four of them binary) and 34 continuous features. The complete feature set is given in Table 3.

The categorical features pose a challenge in algorithms such as ANNs. In this work, the binary features were unchanged. The other three categorical features were converted to numerical features by using a simple label encoding scheme. Label encoding is used over one-hot encoding to preserve the number of input features and to avoid making the data sparse. In order to make sure that the features are considered in the same ranges, the input features were standardized to a zero mean

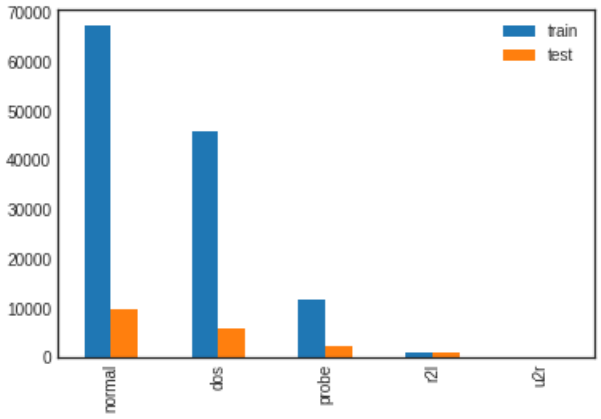


Fig. 9. NSL-KDD: Data distribution across classes. R2L and U2R classes are significantly underrepresented

and unit variance distribution.

4.3.2.1 Classification Results

Using the dataset, two different test-cases were considered to demonstrate the presented explanation methodology. First, a binary classifier was implemented to classify between normal and intrusion data, where all DoS and Probe instances were relabeled as intrusions. Second, a multi-class classifier was implemented to classify instances into three classes—normal, DoS and Probe.

Table 4 shows the classification accuracy achieved by the NN-ADSs for all test cases. It was observed that the FFNN based NN-ADS was able to classify the data with relatively high accuracy levels. It should be noted that these results are presented for completeness. The explanation process is not hinged on the achieved accuracy. In this work, the NN-ADSs are assumed to be sufficiently accurate—in terms of the accuracy scores—to proceed to the explanation stage. In other words, if a NN-ADS does not meet the desired classification accuracy bar, the explanation process could be used as a diagnostics tool rather than a trust building tool.

Table 3. Complete set of features of the KDD-NSL Dataset

No.	Feature Name	Description
1	Duration	Time length of connection
2	Protocol type	Protocol used
3	Service	Destination network service used
4	Flag	Connection Status of the connection
5	Src_bytes	Bytes transferred in a single connection
6	Dst_bytes	Data bytes in a single connection
7	Land	Is the source and destination port numbers same
8	Wrong fragment	Number of total fragments in the connection
9	Urgent	urgent packets in the connection
10	Hot	Number of Hot indicators (e.g. entering a system directory)
11	Number of failed logins	failed login attempts
12	Logged in	Login status: Is logged in
13	Num_compromised	Compromised conditions
14	Root_shell	1 if root shell is obtained, 0 otherwise
15	Su_attempted	Is superuser attempted
16	Num_root	operations performed as root
17	Num_file_creations	Number of file creations
18	Num_shells	Number of shell prompts
19	Num_access_files	operations on access controlled files
20	Num_outbound_cmds	outbound commands in an ftp session
21	Is_hot_login	is the login in the hot list (e.g. root/admin)
22	Is_guest_login	Whether the login is a guest
23	Count	No. of connections to the same dest as current in the past two seconds
24	Srv_count	No. of connections to the same port as the current in the last two seconds
25	Serror_rate	connections with flags S0-S3 among the connections in count (23)
26	Srv_error_rate	connections with flags S0-S3 among connections in Srv_count (24)
27	Rerror_rate	connections with Flag REJ among the connections in count (23)
28	Srv_error_rate	connections with flag REJ among connections in srv_count (24)
29	Same_srv_rate	connections to the same service among connections in count (23)
30	Diff_srv_rate	connections to diff services among the connections in count (23)
31	Srv_diff_host_rate	connetions to different dest among the connections in srv_count(24)
32	Dst_host_count	Number of connections with the same destination IP
33	Dst_host_srv_count	Connections with the same port
34	Dst_host_same_srv_rate	Connections to the same service among the ones in dst_host_count (32)
35	Dst_host_diff_srv_rate	Connections to different services among the ones in dst_host_count (32)
36	Dst_host_same_src_port_rate	Connections to the same source port among the ones in dst_host_srv_count (33)
37	Dst_host_same_diff_host_rate	Connections with diff destinations, among ones in dst_host_srv_count (33)
38	Dst_host_serror_rate	Connections with flag S0-S3 among ones in dst_host_count (32)
39	Dst_host_srv_serror_rate	Connections with flag S0-S3 among ones in dst_host_srv_count (33)
40	Dst_host_rerror_rate	Connections with Flag REJ among ones in dst_host_count (32)
41	Dst_host_srv_rerror_rate	Connections with Flag REJ among ones in dst_host_srv_count (33)

Table 4. Accuracy NN-ADS implemented for NSL-KDD test cases

Classifier	Train Accuracy	Test Accuracy
Intrusion vs Normal	99.38	97.34
DoS vs Probe vs Normal	99.2	94.72

4.3.2.2 Derived Explanations

This section presents the explanations derived for each of the test cases. It should be noted that the explanations are presented with a focus on the anomaly concepts in their respective test cases. First, the IF-THEN linguistic summaries and their quality measures are presented. Then, example concept descriptions are presented using the *high-quality* IF-THEN linguistic summaries.

Explanations: Binary Classification–Intrusion vs Normal

Linguistic descriptions were generated for the concept 'Intrusion'. Table 5 shows the linguistic summaries derived from the NN-ADS system for the 'Intrusion' concept. In this work, single-antecedent-single-consequent linguistic summaries were derived. In order to filter the linguistic summaries, d_t and d_c thresholds were set as 0.9. These linguistic summaries can be used to generate a concept description for 'Intrusion'. If the 'highly-influential' linguistic summaries are used to generate the concept description, the concept description for '**Intrusion**' can be expressed as follows:

—**High** values for *count*, *rerror_rate*, *dst_host_serror_rate*, *dst_host_srv_serror_rate*, *serror_rate*, *srv_error_rate* and a **Low** value for *same_srv_rate* are considered as evidence for an 'Intrusion' by the NN-ADS

Explanations: Multi-class Classification: DoS vs Probe vs Normal

Table 5. Top 10 Linguistic descriptions for NSL-KDD dataset, 'Intrusion' concept

Feature		**Influence**	d_t	d_c
Name	**Value**			
count	high	highly influential	0.98	1.00
rerror_rate	high	highly influential	0.97	1.00
dst_host_serror_rate	high	highly influential	0.96	1.00
same_srv_rate	low	highly influential	0.96	1.00
dst_host_srv_serror_rate	high	highly influential	0.92	1.00
serror_rate	high	highly influential	0.92	1.00
srv_error_rate	high	highly influential	0.90	1.00
hot	low	influential	0.99	1.00
srv_rerror_rate	high	influential	0.99	1.00
dst_host_count	high	influential	0.98	1.00

For DoS attacks, it was observed that the medium to high relevance was given to host based traffic features and time related features when the values of those were high.

> —**Medium and High** values for *count*, **High** values for *dst_host_rerror_rate*, *srv_error_rate*, *dst_host_srv_serror_rate*, *dst_host_serror_rate* and **Low** values for *duration src_bytes dst_bytes land* are considered as evidence for an 'DoS' by the NN-ADS

For Probe attacks, it was noticed that time related features and host based traffic features were being considered for detection by the NN-ADS. In the two highest confidence linguistic summaries that indicate high influence, it can be seen that the NN-ADS is prioritizing traffic that are going out to different destinations or services.

Table 6. Top 10 Linguistic descriptions for NSL-KDD dataset, 'DoS' concept in multi class classification

Feature		**Influence**	d_t	d_c
Name	**Value**			
count	high	highly_influential	0.9986	1.0000
dst_host_rerror_rate	high	highly_influential	0.9875	1.0000
srv_error_rate	high	highly_influential	0.9753	1.0000
dst_host_srv_serror_rate	high	highly_influential	0.9420	1.0000
count	med	highly_influential	0.9255	1.0000
dst_host_serror_rate	high	highly_influential	0.9093	1.0000
duration	low	influential	1.0000	1.0000
src_bytes	low	influential	1.0000	1.0000
dst_bytes	low	influential	1.0000	1.0000
land	low	influential	1.0000	1.0000

Table 7. Top 10 Linguistic descriptions for NSL-KDD dataset, 'DoS' concept in multi class classification

Feature		**Influence**	d_t	d_c
Name	**Value**			
dst_host_rerror_rate	high	highly influential	0.98	1.00
rerror_rate	high	highly influential	0.97	1.00
dst_host_serror_rate	high	highly influential	0.96	1.00
same_srv_rate	low	highly influential	0.96	1.00
dst_host_srv_serror_rate	high	highly influential	0.92	1.00
serror_rate	high	highly influential	0.92	1.00
srv_error_rate	high	highly influential	0.90	1.00
hot	low	influential	0.99	1.00
srv_rerror_rate	high	influential	0.99	1.00
dst_host_count	high	influential	0.98	1.00

—**High** values for *dst_host_rerror_rate*, *rerror_rate*, *dst_host_serror_rate*, *dst_host_srv_serror_rate*, *serror_rate srv_error_rate srv_rerror_rate dst_host_count* and **Low** values for *hot same_srv_rate* are considered as evidence for an 'Probe' by the NN-ADS

4.3.2.3 Adversarial Validation

Once the top feature behavior were identified, the proposed adversarial approach was used to validate them. As mentioned, the identified feature behavior for 'Intrusion' was simulated in the data records classified as 'Normal' to create the adversarial examples. In addition, for performance comparison, the complement of the identified feature behavior was simulated in the data classified as 'Normal'. Therefore three cases were created: 1) simulating low in low feature, med in medium features and high in high features, 2) simulating medium in all, 3) simulating high in low features, and low in high features. It should be noted that these values are simulated in the features identified by the linguistic summaries.

As an example, Figure 10 shows a 2D projection of the test samples from NSL-KDD dataset. The figure depicts samples classified as normal, samples classified as intrusion, and the perturbed adversarial examples created only using the 'highly influential' linguistic summaries. It can be observed that the perturbed normal samples are among the samples classified as intrusions. The 2D projection was obtained using the t-Distributed Stochastic Neighborhood Embedding (TSNE) method.

Further, to verify the relevance of perturbed input features, the relevance scores of the input features were observed for the adversarial samples. Figure 11 shows the input feature relevance scores for a random adversarial sample created using 'highly influential' linguistic summaries. The highlighted features are the perturbed features. The adversarial sample successfully 'tricked' the classifier and the perturbed input

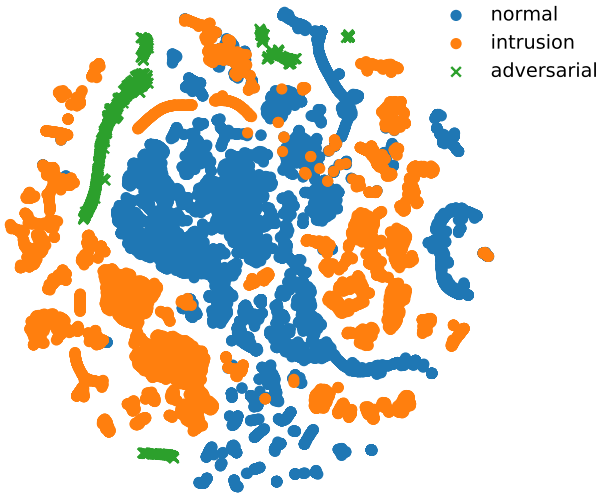


Fig. 10. NSL-KDD: TSNE projection of adversarial examples created using 'highly influential' linguistic summaries

features were the most relevant features. To observe relevance over a large sample, a randomly sampled 1000 records were used. Figure 12 shows the average local relevance of each input feature across the 1000 samples. It can be observed that the perturbed features (highlighted) are consistently supporting the detection of an intrusion. That means, not only the NN-ADS flips the classification label, but also uses the perturbed features to base its decision.

Further, to observe the effect of using different sets of linguistic summaries to adversarial samples, adversarial samples were created by incrementally adding linguistic summaries. Linguistic summaries were sorted by their level of influence, d_t, and d_c, respectively. 5% of high quality linguistic summaries were added to the adversarial examples in each step. At each step, a 1000 instances were randomly sampled and the percentage of labels flipped to 'Intrusion' was observed. In this experiment, the three types of adversarial examples were considered. For each test case, at each level, the percentage of adversarial examples that tricked the NN-ADS was observed. Figure

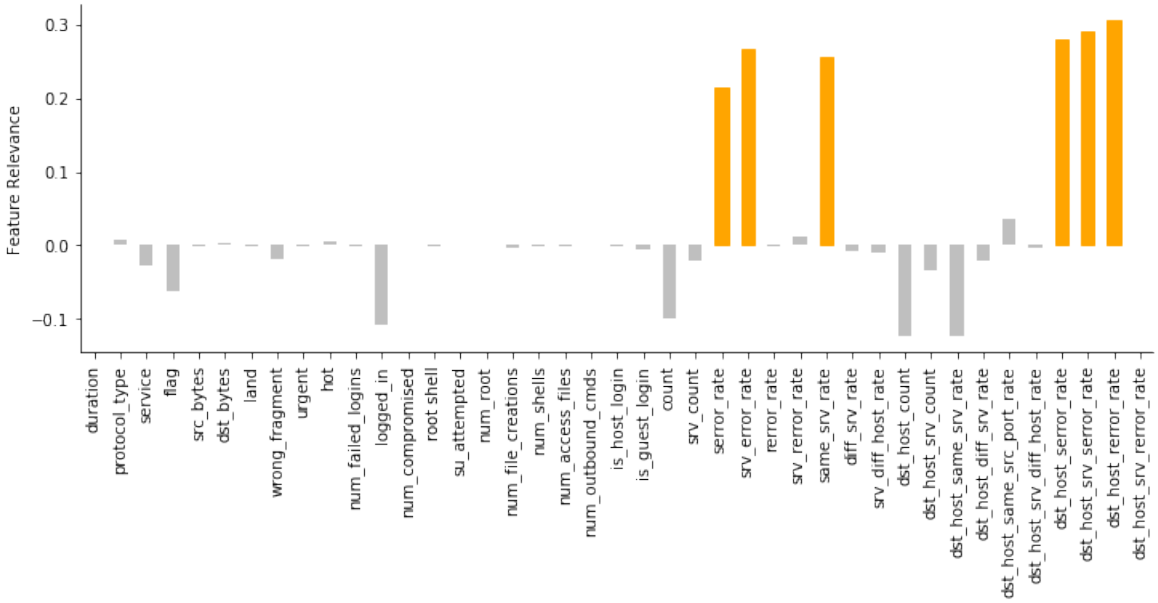


Fig. 11. Feature relevance of a randomly sampled perturbed 'normal' record (originally classified as 'normal') of the NSL-KDD dataset. After the perturbation, the instance was classified as an 'Attack'. It can be seen that the NN-ADS is classifying the instance as an attack based on the features that were perturbed (highlighted ones)

13 shows the results for the three test cases.

4.3.3 CICDS2017 Experiment

CICDS2017 dataset was created by the Canadian Institute of Cybersecutiy as a dataset for comparing different IDS algorithms. The dataset contains communication records for several attack types and normal communication collected over five days. In this work, in order to observe the explanations of a specific intrusion type, Distributed Denial-of-Service (DDoS) was selected. Therefore, the classifier performed binary classification between 'normal' and DDoS. CICDS2017 dataset contains 78 input features and the explanations were derived with respect to all features.

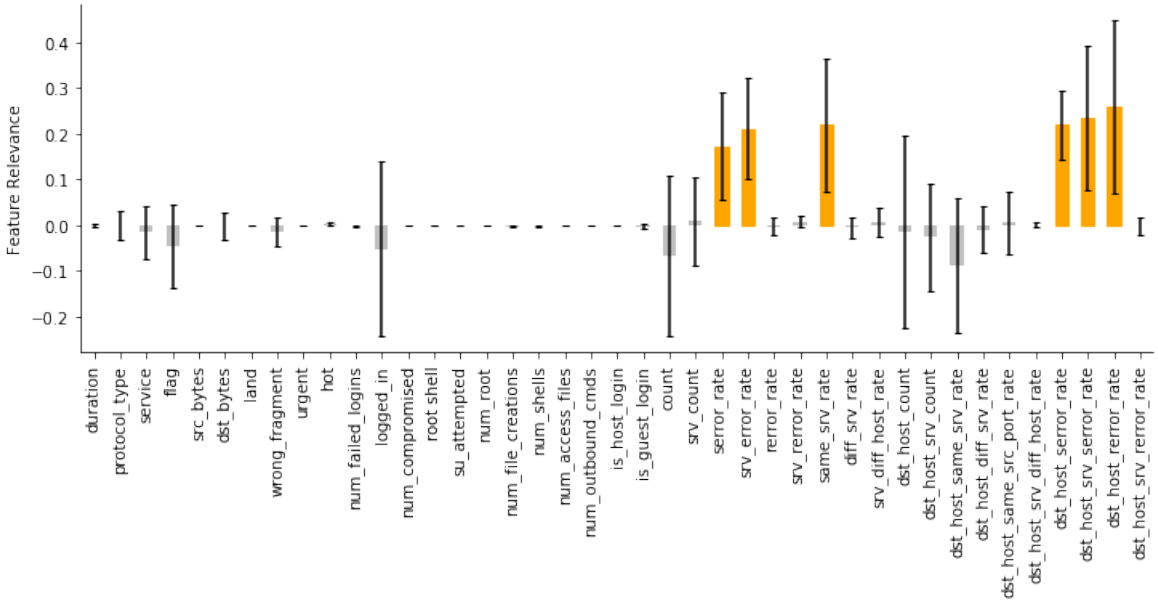


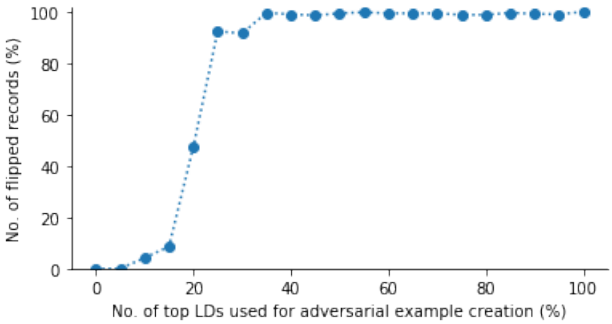
Fig. 12. Average feature relevance of 1000 randomly sampled and perturbed 'Normal' records of NSL-KDD dataset. From the error bars it can be seen that the perturbed features are always positively influencing the decision toward the detection of an attack. 99.2% of the instances were classified as 'Intrusion'

Table 8. Accuracy NN-ADS implemented for the CICDS2017 test cases

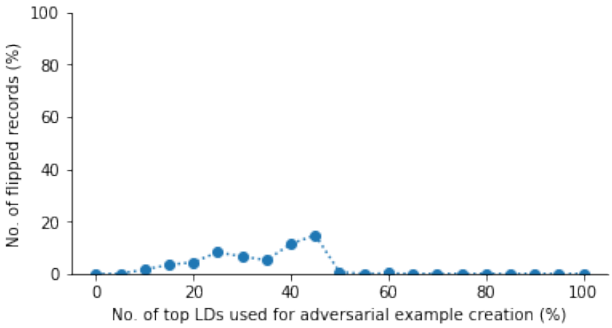
Classifier	Train Accuracy	Test Accuracy
DDoS vs Normal	99.38	98.71

4.3.3.1 Classification Results

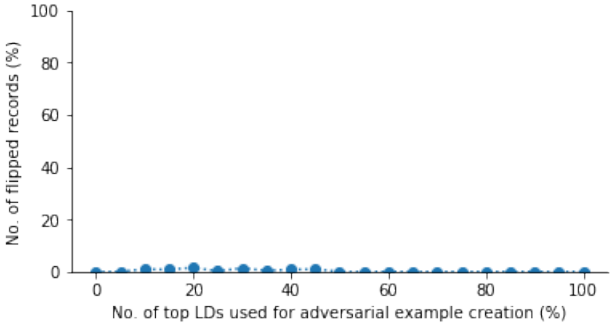
As with the NSL-KDD experiment, the classification results are reported for completeness. It was observed that the NN-ADS was able to successfully classify the data set with a held out test data accuracy of 98.71%. Table 8 shows the training and testing accuracy scores achieved in the CICDS2017 experiment.



(a)



(b)



(c)

Fig. 13. Flipping of classification labels with adversarial samples created with different number of linguistic summaries for NSL-KDD Dataset (a) Using values of LD fuzzy sets, (b) using medium value for all linguistic summaries, (c) switching 'low' and 'high' values in linguistic summaries

Table 9. Top 10 Linguistic descriptions for CICDS2017 dataset, 'DDoS' concept

Feature		Influence	d_t	d_c
Name	**Value**			
Packet_Length_Std	high	highly influential	0.99	1.00
Avg_Bwd_Segment_Size	high	highly influential	0.99	1.00
Bwd_Packet_Length_Mean	high	highly influential	0.98	1.00
Bwd_Packet_Length_Std	high	highly influential	0.98	1.00
Idle_Max	high	highly influential	0.96	0.91
Bwd_Packet_Length_Max	high	highly influential	0.95	1.00
Max_Packet_Length	high	highly influential	0.91	1.00
Subflow_Bwd_Bytes	high	influential	1.00	0.98
Total_Length_of_Bwd_Packets	high	influential	1.00	1.00
Flow_Duration	high	influential	1.00	0.92

4.3.3.2 Explanation and Validation Results

In CICDS2017, the classification was carried out between 'Normal' and 'DDoS'. Therefore, linguistic summaries were derived for 'DDoS'. Table 9 lists the top 10 derived linguistic summaries. Similar to NSL-KDD, the threshold for filtering linguistic summaries were used as 0.9. In the interest of brevity, only the top 10 linguistic summaries are shown. However, there were 30 linguistic summaries that satisfied the thresholds. For adversarial testing, the complete set of linguistic summaries above the thresholds was used. Therefore, similarly to NSL-KDD, a concept description can be generated for 'DDoS'. Using the highly influential linguistic summaries, the concept description can be expressed as follows.

Intrusion Concept Description : **High** values for *Pkt_Len_Std, Avg_Bwd_Seg_Size,*

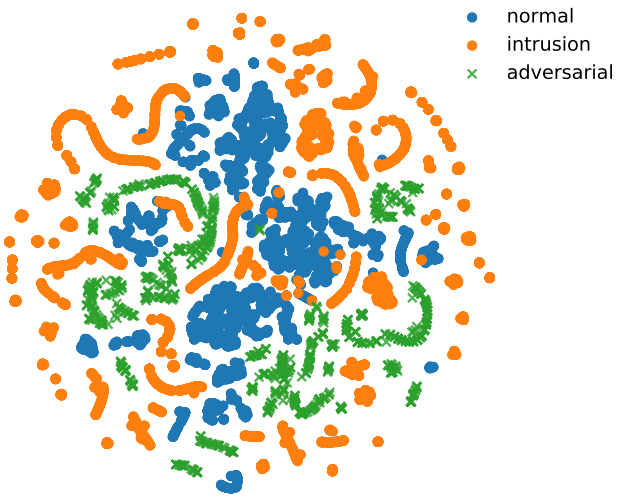


Fig. 14. CICDS2017: TSNE projection of adversarial examples created using 'highly influential' linguistic summaries

Bwd_Pkt_Len_Mean, Bwd_Pkt_Len_Std, Idle_Max, Bwd_Pkt_Len_Max, Max_Pkt_Len are considered as evidence to detect a 'DDoS' attack by the NN-ADS

4.3.3.3 Adversarial Validation

Similar to NSL-KDD, the linguistic summaries were validated using adversarial examples. Relevant feature behavior for DDoS was simulated in 'normal'. Figure 14 shows a TSNE projection of CICDS test samples. Records classified as 'Normal', records classified as DDoS and the created adversarial samples using only the highly influential linguistic summaries are shown in the projection. In CICDS, the data didn't show clear groupings as it did in NSL-KDD. However, amidst the scatter, it can be observed that the adversarial samples tend to be closer to DDoS records, than to 'normal' records.

Furthermore, to verify the relevance of perturbed input features, local relevance scores of adversarial records were observed. Figure 15 shows the relevance of one ran-

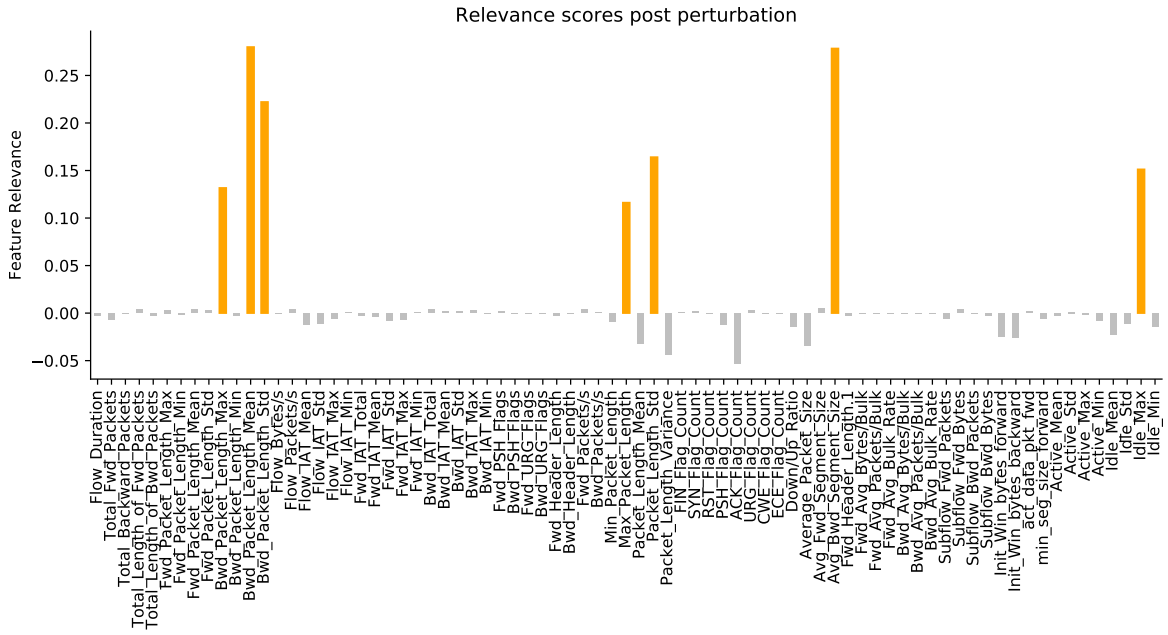


Fig. 15. Feature relevance of a randomly sampled perturbed 'normal' record for the CICDS2017. The instance was classified as an 'Attack' afer the perturbations. It can be seen that the NN-ADS perturbed features are influencing the the decision of the NN-ADS. (highlighted ones)

domly sampled adversarial record. The perturbed input features (highlighted) were the features with highest relevance. Figure 16 shows the mean relevance (standard deviation as error bars) for a 1000 randomly sampled adversarial records. It can be seen that perturbed input features consistently support the detection of a 'DDoS' attack, empirically proving the validity of the linguistic summaries.

Similarly to NSL-KDD, an incremental analysis was carried out to observe the effect of different linguistic summaries. From the 30 linguistic summaries, starting from 5%, the number of linguistic summaries used for adversarial examples was incremented by 5%. At each step, 1000 random adversarial samples were created and the percentage of success was observed. This process was repeated for all three types of adversarial records. The Figure 17 shows the percentage of success at each step

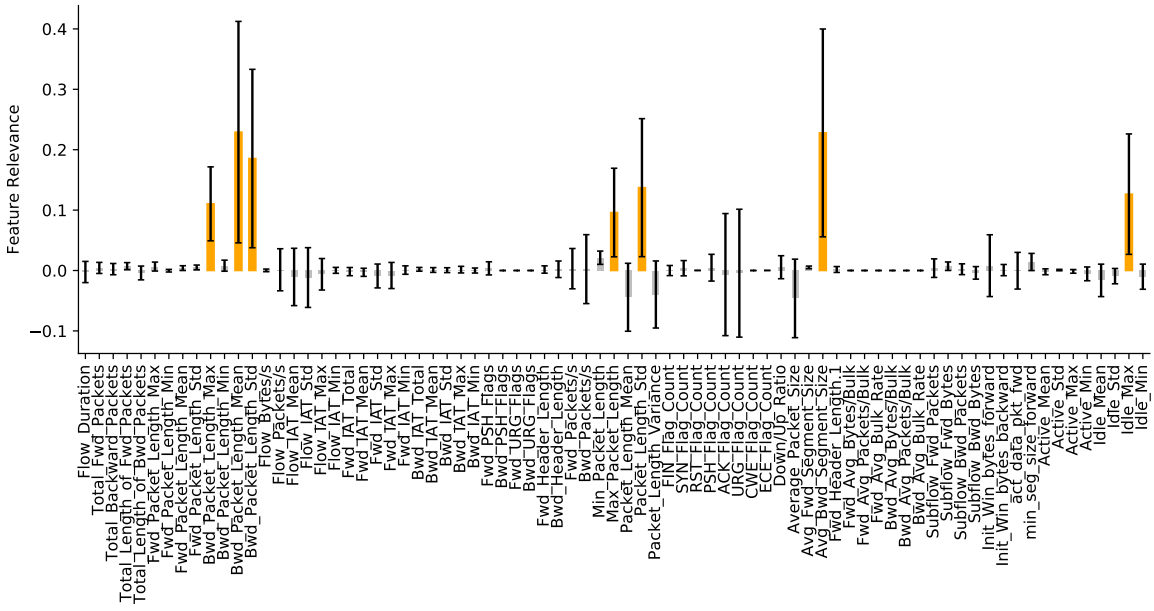
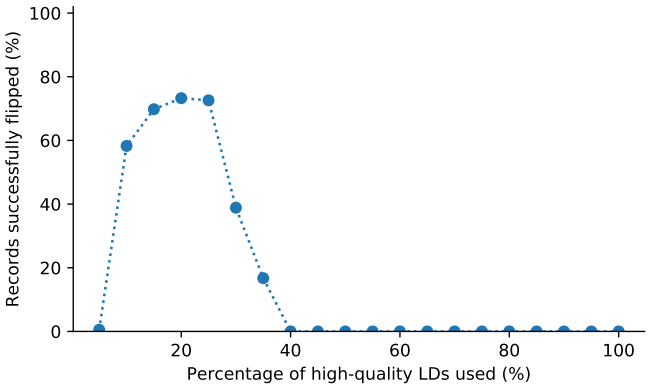


Fig. 16. Average feature relevance of 1000 randomly sampled and perturbed 'Normal' records of CICDS2017. From the error bars it can be seen that the perturbed features are always positively influencing the decision toward the detection of an attack

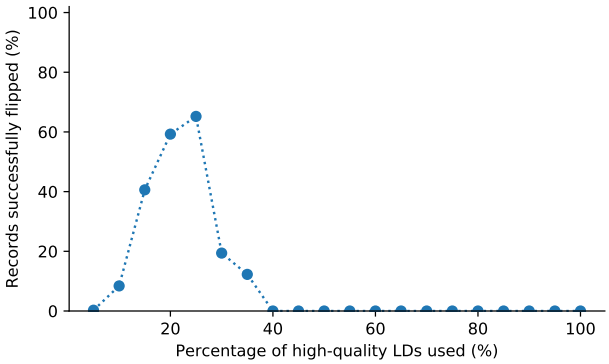
for the three cases. It was observed that the highest success percentage was achieved when all 'highly influential' linguistic summaries were used. Then interestingly, once the others were added, the success percentage reduced to 0%. This needs further analysis and implications are discussed in the next subsection. Further, for Case II, a similar trend was observed but with a lower peak. This is more indication that when the highly influential input feature value increase, the probability of an DDoS attack increases. For Case III, it can be seen that the success rate is 0% throughout. These results are empirical evidence that the derived linguistic summaries are valid.

4.4 Discussion

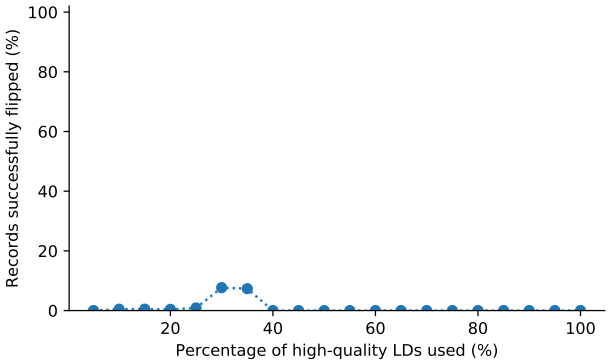
This section discusses the features of presented methodology, its limitations and the ways to extend the functionality. First, the methodology is compared against the



(a)



(b)



(c)

Fig. 17. Flipping of classification labels with adversarial samples created with different number of linguistic summaries for CICDS 2017 Dataset (a) Using values of LD fuzzy sets, (b) using medium value for all linguistic summaries, (c) switching 'low' and 'high' values in linguistic summaries

requirements identified in Chapter 3. Next, the limitations of the methodology and applicable use-cases beyond CP-ADSs are discussed. Then, methods to extend and scale the presented methodology is presented.

4.4.1 Evaluation of the Explanation Methodology

The presented methodology was evaluated against the requirements presented in Chapter 3. Table 10 summarizes the evaluation.

In terms of the types of explanations, the presented methodology only generates overall model explanations. As per individual prediction explanations, existing saliency mapping (e.g. LRP) techniques can be used to explain individual decisions. However, those methods should be customized to generate easily understood explanations given the time sensitivity in this domain.

In terms of the human friendliness of the explanations, the presented methodology generates explanations that consist of input features of the CP-ADS. Therefore, the explanations are tied to the real-system behavior. The complexity of the explanations depends on the number of antecedents in the IF-THEN linguistic summaries generated. The higher the number of antecedents, the precision of summaries increases and completeness of the explanations increases. However, at the same time, the comprehensibility of the summaries decreases. In this presented method, the user can control the number of antecedents in the generated summaries. Therefore, the user can control the balance between interpretability and completeness by controlling the number of antecedents in the IF-THEN summaries. However, the generated explanations are not contrastive. Therefore, to make the summaries completely human-friendly, the explanations need to be made contrastive.

The explanations are generated in textual form and therefore communicated in an understandable form. In terms of evaluation, a quantitative evaluation methodol-

Table 10. Evaluating the presented supervised explainable CP-ADS methodology against the identified key-requirements

Requirements		**This Methodology...**
Type of Exp.	Overall Model	Successfully generated
	Individual Predictions	Future work
Human-friendliness	Real-system behavior	Reflects the real-system
	Contrastive	Future work
	interpretability vs. completeness	User can adjust complexity
Exp. medium	Textual or visual	Textual
Evaluation	Quantitative	A methodology presented
	Qualitative	Future work

ogy is presented. The presented methodology used systematically introduced input perturbations based on the generated explanations and an accuracy score is obtained from the CP-ADS for the perturbed input instances. The accuracy score can be used as a quantitative measure of correctness of the explanations. However, the presented methodology does not include a qualitative evaluation methodology that includes the end-users. A strategy should be developed to involve end-users in the evaluation process to determine how much the generated explanations help the users trust the trained CP-ADS.

In addition to the above features, it should be noted that the presented methodology does not extend to explaining any classification based task in the real-world. For instance, this methodology cannot be applied directly to explain image classifiers at pixel-level. In order to use this method in such a problem, the features will need to be in a higher order, so that the position of the object of interest is consistent in the input space. Therefore, the presented method is not suitable for generating linguistic

explanations in domains such as computer vision.

4.4.2 Extending the methodology

When considering the scalability of the system, in complex real-world scenarios CPSs typically consist of very high dimensional data (E.g. sensor readings from a thermo-chemical plant). In such cases, explanations based on individual features can be incomprehensible. As a solution, input features can be grouped into categories (or a hierarchy of categories) as a preprocessing step. Then, the explanations can be derived using the input feature categories instead of using individual features. The grouping of features are highly domain dependent and the end-user could be heavily involved.

A qualitative evaluation strategy is essential for deploying this methodology in the field. The goal of the evaluation process is to identify whether the presented explanation methodology actually help the end-users trust the trained CP-ADS. This could be achieved by running randomized control trials with a end-user group. One or multiple questionnaires could be designed to gauge the end-users' trust in the system. One portion of the user group can be shown the system and it in action in a simulated environment without the explanations and asked to answer the questionnaire, whereas the other group would be given the system, and the explanations. The results of the questionnaire(s) could gauge the effectiveness of the derived explanations. Further, these evaluations can help expand the list of desired requirements and identify problem/user specific requirements.

4.5 Conclusions and Future Work

This chapter presented a methodology for generating explanations of a trained, supervised ANN-based CP-ADS—referred to as NN-ADS. The methodology focused

on deriving explanations about what the ANN had learned about each anomaly-type (concept) in its training phase. The derived explanations were presented to the user in terms of linguistic concept descriptions. In addition to the explanation methodology, this chapter presented a methodology for validating the generated explanations. Input instances were intentionally perturbed using the derived explanations and the NN-ADS's responses to those perturbed examples were used to validate the system. The methodologies were tested in several experiments. The results from the validation step empirically verified the validity of the generated explanations. Furthermore, the presented explainable CP-ADS system was evaluated against the requirements identified in Chapter 3 and the limitations of the method were discussed. Three main limitations were identified: 1) individual prediction explanations are not generated, 2) explanations are not contrastive, 3) no qualitative evaluation strategy designed. Further, it was identified that the presented method would create too complex explanations for very-high dimensional spaces, and the methodology will not apply to domains where the object of interest is not static in the feature space, e.g. computer vision. Methodologies were proposed to generate contrastive explanations and to adapt explanations to very-high dimensional spaces as future work.

CHAPTER 5

EXPLAINING UNSUPERVISED NEURAL NETWORKS BASED ANOMALY DETECTION

In general, pattern recognition tasks are dominated by supervised learning algorithms [84], [85]. However, the main drawback of these state-of-the-art pattern recognition algorithms is that they depend on the availability of large labeled datasets. The scarcity of labeled data in the real world, due to the cost of acquiring sufficiently large datasets is a major hurdle to deploy supervised ANNs in the real world [86], [87]. Therefore, any suite of algorithms aimed at the industry, should consider unsupervised learning techniques to leverage abundantly available unlabeled data [87], [88], [89]. CPSs generate massive amounts of data are and unsupervised learning techniques can be used to learn behavioral patterns in CPSs.

Unsupervised learning in neural networks is mainly used to learn better representation from raw data [90]. Learning happens through non-linear approximations between raw data and their reconstructions using methods such as auto-encoders [91]. One of the unsupervised neural network algorithms that have proven its usability in pattern recognition is Self-Organizing Maps (SOMs) [92]. SOMs have been used in a multitude of real-world applications for learning patterns in unlabeled data [93], [94].

However, the main drawback of the SOM is its inability to learn features with multiple layers of abstraction like ANNs [95], [96], [97]. In this work, a multi-layered SOM architecture, named Deep Self-Organizing Maps (DSOM) is presented for learning CPS behavioral patterns. The main advantage of the presented algorithm is that it adds the capability of high-level feature abstraction to the single layer SOM. This

chapter presents the algorithm details, how to explain the DSOM and experimental details on its pattern recognition capability and explanation capability.

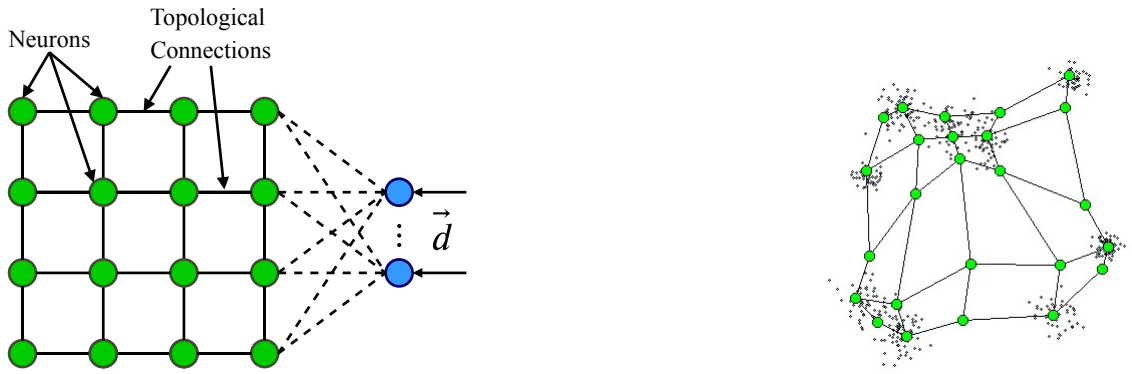
This chapter first introduces single layer SOMs and the learning algorithm of a SOM. Second, the idea of DSOM, its architecture and the algorithm for learning the behavioral patterns of a CPS using the DSOM is presented. Third, the explainability of the algorithm is discussed. Next, the methodology for visually and linguistically explaining the patterns learned by the DSOM is presented. Then, experiments and results are shown to demonstrate the pattern recognition and explanation methodologies. Finally, the chapter is concluded with the main findings and possible next steps.

5.1 Self-Organizing Maps

The Self-Organizing Map (SOM) algorithm was developed in by Kohonen [98], [92]. SOM is a neural network algorithm that employs unsupervised learning to re-arrange (re-organize) its structure to mimic the topological properties of data. It is frequently used as a dimensionality reduction and a data compression tool. The SOM is a very popular technique for exploratory data analysis [99], [100]. The SOM is capable of mapping high-dimensional data distributions onto low-dimensional distributions while preserving the most important topological relationships of input data [101]. SOMs have been used in various engineering applications for anomaly detection [93], [101], system identification [102], [103], time-series analysis [94], [104], [105], speech recognition, robotics and process control [103, 106], [107–113]

5.1.1 Training Self-Organizing Maps

SOM uses unsupervised "winner-takes-all" (WTA) competitive learning and co-operative adaptation to adjust itself to the topological properties of the input dataset.



(a) (b)

Fig. 18. Self-Organizing Map displayed in the output space (a), and in the input space adapted to a 2D distribution of data points (b)

The SOM consists of a topological grid of neurons typically arranged in 1D or 2D lattice [114]. Each neuron has a position in input space as well as a fixed position in output space. The fixed grid defines the spatial neighborhood of each neuron. Each neuron maintains a synaptic weight vector $\vec{w}$, with the dimensionality of the input space. The input dataset consists of input patterns that can be denoted as $\vec{d}$. The structure of a 2D SOM is depicted in Figure 18. First, all neurons are first randomly initialized and then iteratively adapted based on the training set of input data. The training process can be described in several steps as follows [114]:

Step 1 – Initialization: Randomly initialize all synaptic weight vectors of the neurons.

Step 2 – Sampling: Select a random input pattern from the training dataset.

Step 3 – Competitive Learning: Find the Best Matching Unit (BMU) for the current input instance $\vec{d}$. The BMU the neuron with the minimum Euclidean

distance between its synaptic weight vector and the $\vec{w}$:

$$BMU(\vec{d}) = \underset{k}{argmin} \|\vec{(}d) - \vec{w}_k\|, \quad k = 1, 2, \ldots, K \tag{5.1}$$

where, $BMU(\vec{d})$ is the BMU of the input instance $\vec{d}$, $\|\cdot\|$ operator denotes the Euclidian distance norm, and K is the number of neurons in the SOM.

Step 4 - Cooperative Updating: Update the synaptic weight vectors of all neurons in SOM using the cooperative update rule:

$$\vec{w}_k(t+1) = \vec{w}_k(t) + \alpha(t) * h_{k,BMU(\vec{d})}(t) * (\vec{d} - \vec{w}_k(t)) \tag{5.2}$$

where, t denotes the current epoch, α is the learning rate and $h_{k,BMU(\vec{d})}$ is the degree of membership of the unit to the defined neighborhood centered at $BMU(\vec{d})$. The neighborhood function is typically implemented as a Gaussian function centered at the selected wining neuron. Its amplitude applied to neuron k is calculated as follows:

$$h_{k,BMU(\vec{d})} = e^{\frac{\|\vec{d}-\vec{w}\|}{2\sigma^2}} \tag{5.3}$$

The neighborhood size is determined by the σ. In order to enforce convergence, the size of neighborhood is reduced by decreasing the parameter σ. Typically, the exponential decay rule is applied. The learning rate controls the rate of adaptation of individual neurons. Like the size of the neighborhood function, its value also exponentially decays with the elapsed training time.

Step 5 Convergence Test: Until a specified convergence criterion is met go to Step 2.

Once the training process is completed, the number of times each neuron k was

selected as the BMU is stored as $N_{BMU,k}$:

$$\sum_{k=1}^{K} N_{BMU,k} = M \tag{5.4}$$

where K is the number of neurons and M is the total number of data points. This value was calculated after the convergence of the SOM.

The relatively simple unsupervised learning algorithm of SOMs together with the visualization and feature reduction capabilities, make them an attractive algorithm for real-world applications, where labeled data are scarce. The ability to mimic the topological properties of the dataset, leads to the SOM producing a generalized representation of the data distribution, i.e. clustering the dataset [92], [115]. Further, the SOM has the capability of mapping a high-dimensional dataset to a low dimensional grid, making SOMs powerful tools for visualizing high dimensional data [103], [97, 116, 117]. Other advantages of SOMs include understandability [118], [106], ease of optimization [119], and better capability of revealing overlapping structures in clusters compared to traditional clustering methods [120]. These capabilities make SOMs a very useful tool for exploratory data analysis.

As mentioned, the major drawback of SOMs is its limited capability of high-level feature abstraction due to the shallow structure [95]. One of the recent attempts at alleviating this limitation was to explore a deep architecture of SOMs, named Deep Self-Organizing Maps (DSOM) by Liu et al. [117]. The authors tested the DSOM on the MNIST dataset and were able to achieve a better classification accuracy compared to the single layer SOM. Since DSOM architecture uses the same learning mechanism as SOMs, it inherits all the advantages of SOMs mentioned above. However, the authors of [117] explored a supervised learning algorithm with DSOM and thus relied on the availability of labeled data. An unsupervised DSOM architecture has the following main advantages: 1) the ability to leverage unlabeled datasets, 2) hierarchical

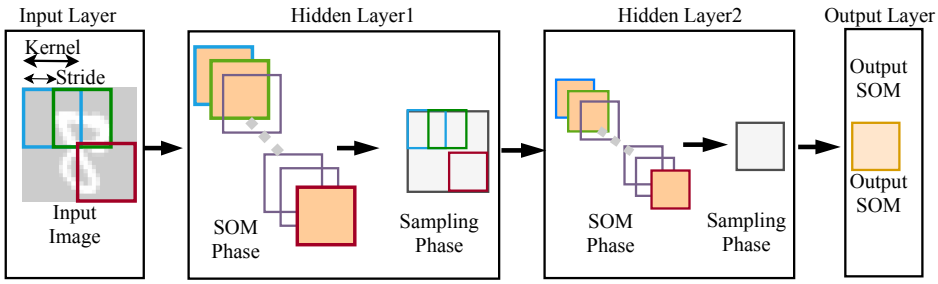


Fig. 19. The multi-layered Deep Self-Organizing Map

feature abstraction based unsupervised learning, and 3) the ability to deploy without special hardware [96].

5.2 Deep Self-Organizing Maps

The concept of Deep Self-Organizing Maps was initially proposed to provide the SOM with high-level feature abstraction capability [117]. The DSOM is a multi-layered architecture which consists of an input layer, hidden layers, and an output layer.

DSOMs were initially designed by merging the concepts of SOMs and Convolution Neural Networks (CNNs). SOMs provide the underlying learning mechanism to DSOM whereas the high-level feature abstraction process in DSOM is inspired by Convolutional Neural Networks (CNNs). In CNNs, in each hidden layer, each unit (neuron) receives inputs from a subset of units in the preceding layer (local receptive fields/patch) [84]. The lower-level features learned in the preceding layer are combined in the current hidden layer to generate higher-level features. This idea was incorporated into the DSOM architecture so that higher-level layers are capable of learning more abstract information than its preceding layer.

We proposed an improved version of the proposed architecture of the initially proposed Deep SOM in our previous work [64, 96, 97]. The learning methodology

was improved in two ways: 1) the learning algorithm was made completely unsupervised and 2) the architecture was modified to learn features of different resolutions in parallel in a single hidden layer. This algorithm was initially proposed for pattern recognition in images [64]. The algorithm was modified to be explainable and usable for the CP-ADS domain. First, the architecture proposed for image pattern recognition is presented. Then, the explainability and modification of algorithm is presented.

5.2.1 Architecture

This section discusses the architecture of the algorithm we presented for image pattern recognition. This is necessary background to discuss the algorithm used for CP-ADS. The DSOM consists of an input layer, hidden layers, and an output layer. It should be noted that, since DSOM uses the notion of the local-receptive fields, inputs are structured as a 2-D image. When using numerical data, such as CPS sensors, the data need to be restructured into a 2-D format to be fed into the DSOM. This can be viewed as a re-sampling of the feature space in the learning process.

Input Layer: Forwards the input images to the DSOM

Hidden Layer: Hidden layer consists of parallel layers. Each parallel layer consists of two phases: 1) SOM phase and 2) sampling phase. In the SOM phase, each input record is segmented into subsets of smaller feature spaces (patch). Then, each patch is sent to its own SOM unit. Each SOM finds the best matching unit for the input patch using the learning algorithm discussed in the previous section.

In the sampling phase, two sampling processes take place. First, a feature map for each parallel layer. In order to create the feature map, BMUs for each image

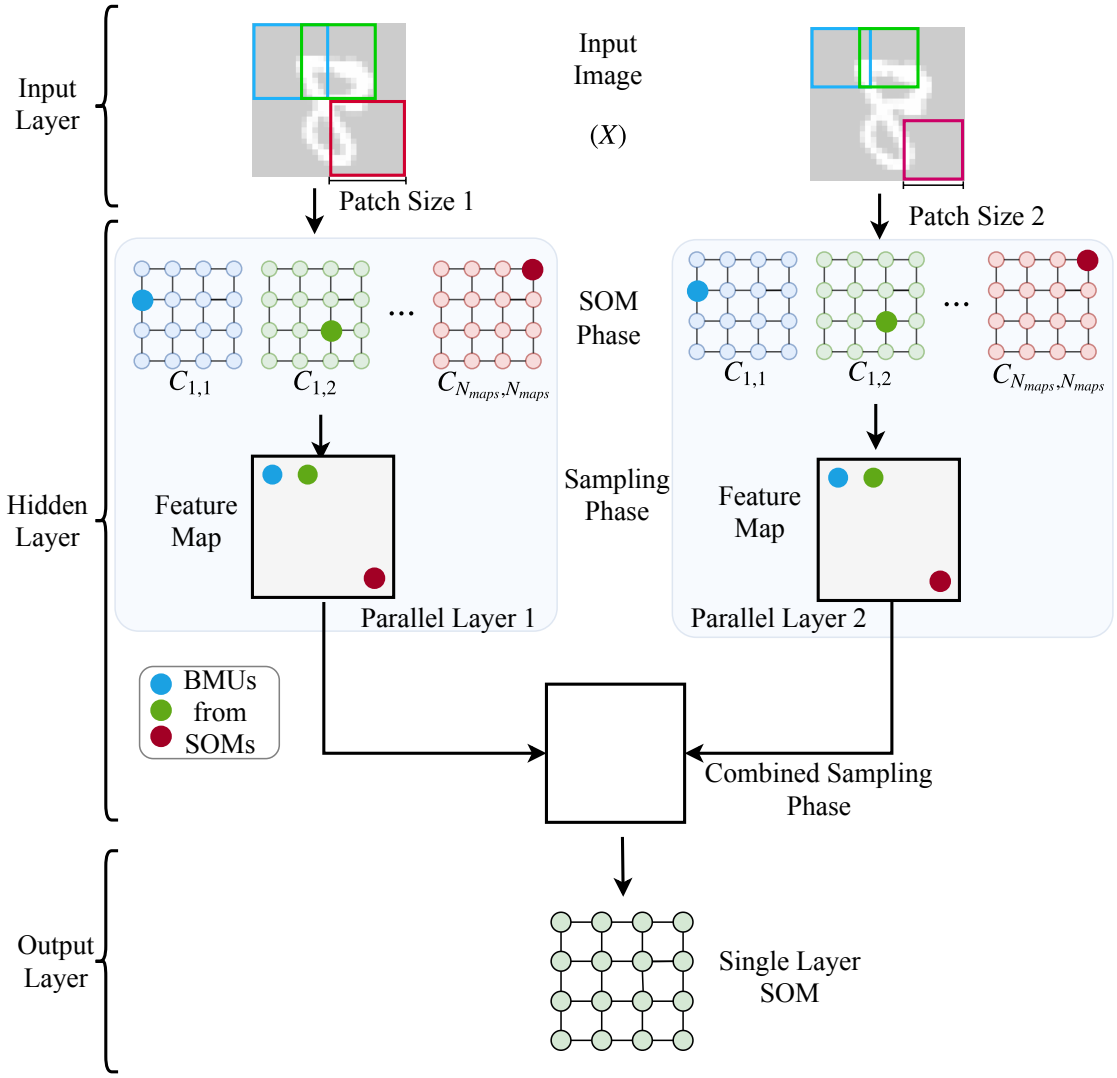


Fig. 20. The proposed architecture of the Deep Self-Organizing Map

patch is combined into a 2D grid. Second, those feature maps are combined to create a single feature map. This single feature map acts as the input for the next hidden layer.

Output Layer: The output layer of the proposed DSOM consists of a single SOM. It receives the output from the sampling phase of the last hidden layer. This representation contains the extracted abstract and pertinent information for classification.

The presented algorithm uses different sized patches (multi-scaled patches) in parallel SOM layers of the hidden layers. It has been shown that multiscale patch approaches can be used to good effect in improving classification accuracy by extracting complementary information in other classification algorithms [121–123]. Figure 20 shows an example DSOM architecture with two parallel layers, where the sizes of patches in the parallel layers are different from each other.

The above modification enables the algorithm to learn feature spaces of different sizes and resolutions by using different map sizes and patch sizes in the parallel layers. We hypothesized that this ability will lead to improved pattern recognition capability and robustness to noise.

5.2.2 Training Algorithm

Training algorithm of the DSOM is presented in Table 11. Similar to the SOM, weights of the network are randomly initialized. In a hidden layer, the SOM phase consists of P parallel SOM layers with P different patch sizes. For each patch size, the number of patches along one dimension is calculated as follows:

$$N_{map} = ceil\Big(\frac{M-K}{S}\Big) + 1 \tag{5.5}$$

where $ceil(\cdot)$ calculates the smallest integer upper, M is the pixel width/height of the input image X ($M * M$ image), K is the width/height of the patch ($K * K$ patch) and S is the stride of the patch. Therefore, for a 2-D input, $N_{map} * N_{map}$ number of patches are created form the input image for each patch size, i.e. $N_{map} * N_{map}$ number of SOMs are created for each parallel layer.

First, in all the parallel SOM layers, the BMU selection for its respective patches is carried out using the SOM learning algorithm discussed earlier in the chapter. Then, the sampling process is carried out for each parallel SOM layer. Therefore, P feature maps are created. The subroutine of the parallel layer is given in Table 12. Then, the P parallel feature maps are combined to create a single feature map. This subroutine is given in Table 13. In this, parallel feature maps are converted into one-dimensional arrays and concatenated into a single array. Then, the resultant array is reshaped to a 2D grid which acts as the input image to the next hidden layer.

After the processing, the hidden layers, the combined feature map generated from the last hidden layer acts as the input to the output SOM layer which consists of a single SOM. The output SOM BMU is found SOM using the SOM learning algorithm. This process is carried out for all the input patterns for a single epoch and is repeated for the desired number of epochs or until a specific convergence criterion is met.

5.3 Explainability of Deep Self-Organizing Maps

This section analyses the explainability of the DSOM algorithm we presented for image classification. If the DSOM is to be adapted to CP-ADS domain, the algorithm needs to be explainable.

As mentioned in Chapter 3, the explanations presented to the user should be presented in terms of the real system, as it's the end-users' domain. Therefore, when input data are processed through multiple layers, there should be a way to trace the

Table 11. Algorithm for training the Deep Self-Organizing Map

Algorithm: DSOM Training

Inputs: Data (X), Number of hidden layers (L), Number of Parallel layers (P)

Output: Trained DSOM

Random Weight initialization
for each epoch e do
    for number of training samples do
        $x \leftarrow$ pick random input record from X
        for each hidden layer l do
            $featureMapList \leftarrow empty\ list\ of\ length\ P$
            for each parallel SOM layer p do
                $featureMapList[p] \leftarrow ParallelLayer(x)$
            end for
            $x \leftarrow CombinedSampling(featureMapList)$
        end for
        OutputSOM $\leftarrow$ Find BMU for x using SOM learning
    end for
end for

Table 12. Subroutine for processing a parallel layer in the Deep Self-Organizing Map

Subroutine: ParallelLayer
Inputs: Input record (x)**, Number of patches** (p)
Output: Sampled feature map
1: $featureMap \leftarrow$ empty list of length p
2: **for** each patch $x^{`}$ do:
3: $index_x \leftarrow$ the location of $x^{`}$ w.r.t. x
4: $BMU_{x^{`}} \leftarrow$ get BMU index for $x^{`}$ on corresponding SOM
5: $featureMap[index] \leftarrow BMU_{index}$
6: **end for**

Table 13. Subroutine for generating the combined sampling feature map in the Deep Self-Organizing Map

Subroutine: CombinedSampling
Inputs: List of feature maps from each parallel layer ($featureMapList$)
Output: Combined feature map
1: $comFeatureMap \leftarrow$ Append $featureMapList$ to a single list
2: $l \leftarrow$ length of $comFeatureMap$
3: **if** $\sqrt{l} \notin \mathcal{N}$ **fo**; $\mathcal{N} = \{1, 2, 3, 4, ...\}$
4: Use zero-padding on $comFeatureMap$ until $\sqrt{l} \in \mathcal{N}$
5: $CFM \leftarrow$ Reshape $comFeatureMap$ to a 2D vector of size $\sqrt{l} * \sqrt{l}$
6: **return** CFM

output back to the inputs. This is especially necessary when abstract features are being learned in the hidden layers. Then, the contribution of each input feature can be quantified, resulting in an explanation of why a certain output is reached by the network.

As mentioned, the goal of explaining in an unsupervised CP-ADS context is to understand the data. In a single layer SOM, the neurons of the SOM acts as a generalized representation of the training-data, where each neuron is a representation of the training instances it was selected as the BMU. Therefore, the 'knowledge' of the SOM is stored in the neuron weights of the SOM. When we draw a parallel to the DSOM, the neurons of the output layer SOM is what contains the generalized representations of the training data. However, unlike the single layer SOM, the DSOM output layer is learning from abstract inputs and neurons represent a compressed dimensional space. Therefore, it is necessary to trace back and map the compressed dimensional space to the real-system data (the original input space).

In the presented DSOM algorithm, the inputs are processed through SOM and sampling phases in each hidden layer. If we recall, the sampling phase takes the indexes of the BMUs in the SOM phase to make up the abstract feature space. This is a non-continuous transfer and it prevents from tracing the abstract features to the previous layer. This is an obstacle in mapping the compressed feature space in the output layer to the original input feature space. Since, it is required to generate explanations in terms of the input feature space, this is an obstacle to explainability.

5.3.1 Modifying the Algorithm for Explainability

The explainability drawback can be fixed by either changing the algorithm to preserve the original dimensionality of data through the layers, or by changing the algorithm to process inputs through the layers with continuous transfer functions. In

the first method, the output SOM contains the dimensionality of the original feature space and thus the neurons can be used as generalized representations of the training data. In the second method, the output SOM neurons are of a compressed feature space but due to the continuous transfer functions in the hidden layer, the compressed feature space can be mapped back to the original feature space.

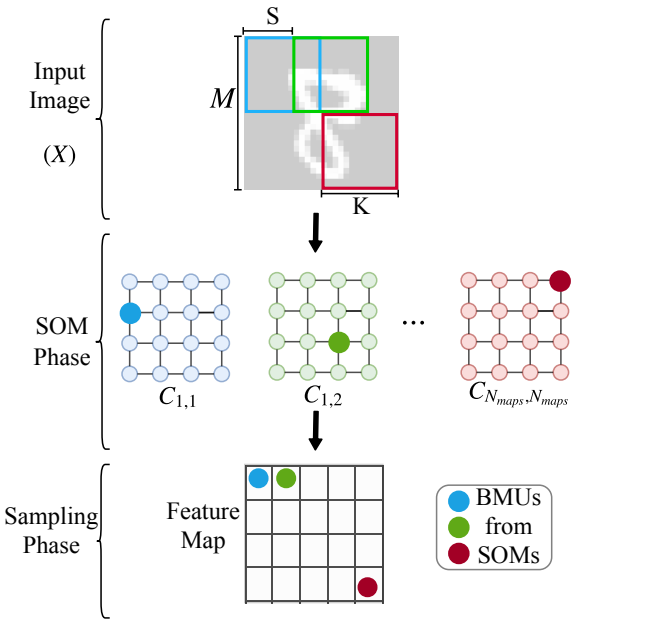
In this work, the first method is implemented and the algorithm is modified to preserve the original dimensionality. The modification is done in the sampling and combined-sampling phases of the algorithm.

Modified Sampling layer: In the modified algorithm, the sampling layer samples the weight vector of the BMU instead of the index. Note that the weight vector of the BMU has the same dimensionality as the input patch that the SOM is processing. Therefore, the BMU weight is a generalized representation of that patch according to the SOM. Once the BMU weight vectors are extracted from all the SOMs, they can be used to reconstruct the original input dimensional space. It should be noted that if the stride is not equal to the patch size, there is an overlap in input patches. This should be considered when reconstructing the space. In this work, the weights of overlapping regions are averaged in the reconstruction step. Figure 21 illustrates the process of the sampling layer before and after the modification.

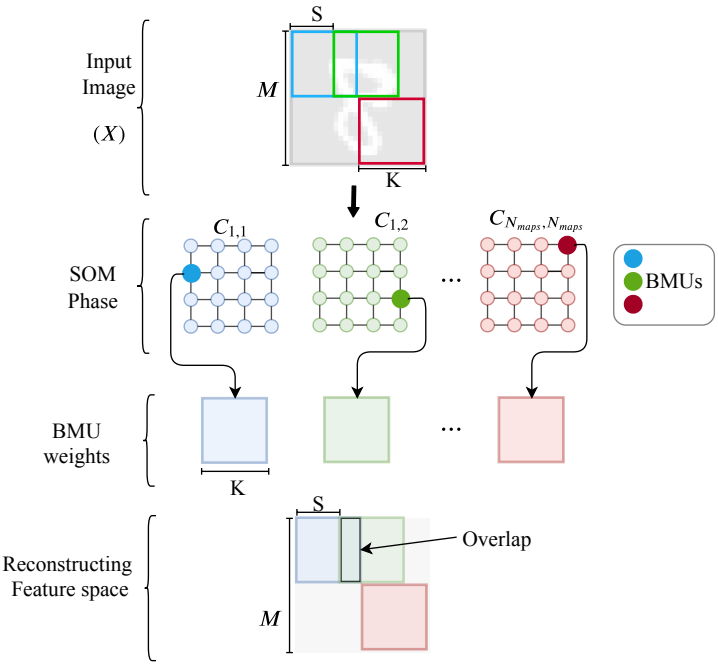
Modified combined sampling layer: Before, the combined sampling layer received P features maps with BMU indexes. After the modification, the combined layer receives P images of the size of the image. In this work, the combined sampled feature map is the average across the P images.

5.4 Explaining Deep Self-Organizing Maps

This section presents the methodologies for explaining the identified Cyber-Physical System behavioral patterns using the DSOM. Explanations are generated in



(a)



(b)

Fig. 21. Modifying the Sampling layer to preserve the dimensionality through the hidden layers for explainability. (a) The sampling layer before the modification, the feature map is the indexes of the BMUs, (b) sampling layer after the modification. The input space is reconstructed using the weights of the BMUs.

two forms: 1) linguistic and 2) visual. The overall explanation framework is given in Figure 22.

First, the unlabeled Cyber-Physical data are used to train a DSOM. In this step, separate DSOMs can be used for cyber data and physical data or a combined data can be used on a single DSOM, or three DSOMs could be deployed to learn cyber behavior, physical behavior, and cyber-physical behavior respectively. In the rest of the description, it is assumed that only one DSOM is trained to learn cyber-physical behavior. Once the DSOM is trained, the trained output SOM is used to explain the learned Cyber-Physical behavior.

Once the DSOM is trained, the output SOM neuron weight vectors can be considered as generalized representations of the training dataset. Therefore, the proposed method identifies each neuron in the output SOM as a data point, thereby compressing the number of data points used for explaining the CPS behavior. Since neurons are a generalized representation of data, using the weight vectors of the neurons to explain the behavior makes the method more robust. However, since some neurons are selected as the BMU more frequently than others, the number of times each neuron was selected as a BMU needs to be considered in the explanation process. Therefore, once the DSOM is trained, the training dataset is processed through the DSOM as an inference step to count the number of times a neuron was selected as the BMU ($N_{BMU,k}$). This value is used as a weight in generating linguistic explanations.

Since the set of weight vectors of output SOM neurons can be used as generalized representations of the dataset, the weight vectors are fed to a traditional clustering algorithm to identify the clusters in the SOM weights. This clustering algorithm can be chosen as seen fit to the application or through cross-validation. For instance, K-Means clustering algorithm can be used as to identify a predefined set of clusters, Fuzzy C-Means algorithm can be used to identify a fuzzified cluster space for the

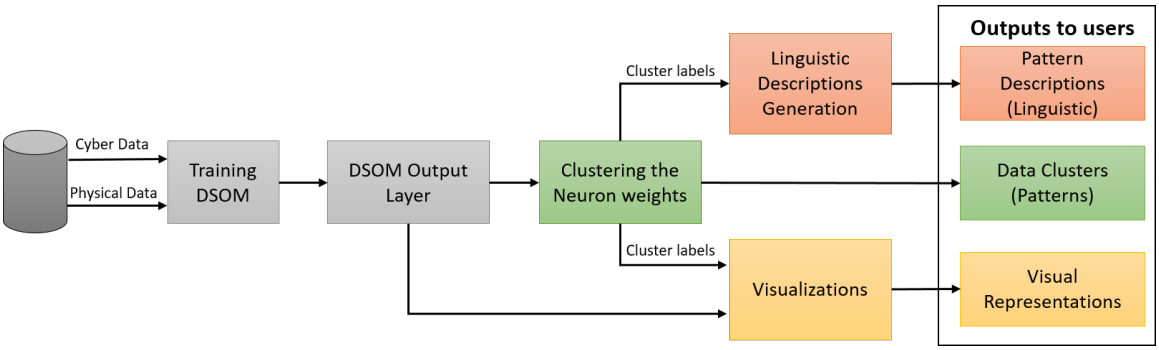


Fig. 22. Framework for explaining Deep Self-Organizing Maps

data, Gaussian Mixture Models can be used to identify a probabilistic cluster space, and methods like DBSCAN can be used for density based clustering. Once the SOM weights are clustered and the cluster distribution across the map is identified, the visual and linguistic explanation of the CPS behavior are generated.

5.4.1 Explaining through Linguistic Descriptions

Similar to the previous chapter, linguistic summarization techniques are used for generating the explanations of the CPS behavior. As mentioned in the previous section, the weight vectors in the output SOM is clustered to assign a label to each neuron. This, in turn, applies labels to every data point based on the label of its BMU. The linguistic summaries are generated with respect to the clusters of neurons.

As with the previous chapter, all the features are fuzzified into a preset number of fuzzy sets. For instance, the same fuzzy set configuration that is given in Figure 8(a) can be used for fuzzification of features. It has to be noticed that the configuration is completely adaptable to the application, feature values and the desired level of precision of the explanations.

For each cluster, Yager linguistic summaries [73] can be used to identify input features that show a clear pattern of behavior in the cluster. This results in a de-

scription of the cluster with respect to individual features with clearly distinguishable behavior. An example Yager type summary can be expressed as follows:

$$Q \ records \ have \ S \ y \ values \tag{5.6}$$

Where Q is a quantifier, S is a property and y is the object. The above explanation can be written in terms of fuzzy sets:

$$S_q \ recods \ have \ S_y \ y \ values \tag{5.7}$$

Where S_q is the fuzzy set of the quantifier and S_y is the fuzzy set of the property. For example, a potential summary of a cluster can look like, *"most data records in the cluster have high thermal capacity values"*. The generated summary with fuzzy sets to represent that would be as follows:

$$most \ datarecords \ have \ high \ "thermal - capacity" \ values \tag{5.8}$$

SOMs tend to interpolate between the map units and thus the neurons in the cluster boundaries can have little to no data points associated with them. In order to take this factor into consideration, the quality assessment of a Yager summary is modified to fit the DSOM based method as follows:

$$T_{yager} = \mu_{S_q}\left[\frac{\sum_{n=1}^{L} \mu_{S_y}(w_{l,y}) \times N_{BMU,l}}{D}\right] \tag{5.9}$$

Where, d_t is the degree of truth of the summary, L is the number of neurons in the DSOM output layer, D is the number of data points in the cluster and, $w_{l,y}$ is the weight of neuron k for dimension i, and $N_{BMU,l}$ is the number of times neuron l was selected as the best matching unit.

5.4.2 Explaining through Visualization

One of the main benefits of the SOM-based methods is the ability to visualize high-dimensional data. The visualizations combined with the linguistic explanations presented in the previous section can be used to explain the Cyber-Physical behavior space. Similar to the linguistic explanations, output layer SOM of the DSOM is used to generate the visualizations. The following visualization techniques can be used to visually explain the DSOM [124]. It should be noted that these techniques are well established techniques to visualize data and are not contributions of this dissertation.

Hitmaps: Hitmap visualizations show the 2D grid of the DSOM output layer with the colors representing the $N_{BMU,k}$ values. Usually, a grayscale map is used to show the dispersion of data across the map. If proper clusters are identified, data should be grouped in certain areas of the map with space in between groups.

Component plane visualizations: Each component plan, i.e. the behavior of each input feature can be visualized with respect to the DSOM. The spread of values of an input feature can be identified using these visualizations. Further, these enable a user to identify correlated input features given the cluster space.

Distance matrices: Distance matrices are popularly used for presenting the clusters in a SOM. The most popular distance matrix type is the U-Matrix [99]. In the U-Matrix, each square of the map contains the average distance to its neighbors. The U-Matrix can be used to observe the separation between the identified clusters using the clustering algorithm.

Manifold Projections: t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction technique [125]. t-SNE can be used to map the

weight vectors of the trained DSOM to a 2D or a 3D space to visually observe the separation of clusters.

5.5 Experimental details: Analysis of Pattern Recognition Capability

This section discusses the details of the experiments conducted on the presented DSOM algorithm to investigate its pattern recognition capability. Since DSOM is a novel algorithm, establishing its pattern recognition capability was important prior to explanation. The experimentation conducted on its explainability is presented in the next section.

The presented DSOM algorithm was tested on several datasets and it was compared against other state-of-the-art unsupervised learning algorithms on the same datasets.

5.5.1 Datasets

Three datasets were used for experimentation: 1) MNIST [126], 2) Gas Sensor Array Drift (GSAD) dataset [127], and 3) Smart Phone dataset for Human Activity Recognition (SP-HAR) [128]. For all datasets, balanced subsets of the data records were selected to alleviate the class imbalance problem.

The **MNIST** dataset contains images of hand-written characters (digits from 0-9), each 28×28 pixels in size. The complete MNIST dataset contains 55000 train images and 10000 test images. We used a significantly smaller training set of 3000 images was used to reduce the classifier training time. The complete testing set (10000 images) was used to test the accuracy of the algorithms.

The **GSAD** dataset contains 13910 records collected from 16 chemical sensors from a gas delivery facility. The dataset contains data about 6 gases collected over 36 months. We considered only the first 21 months were used to avoid concept drift

in data. Since each data record consists of 121 dimensions, each data record was arranged to an 11×11 image.

The **SP-HAR** dataset consists of 10299 smartphone sensor records of 30 subjects performing six different daily living activities. A balanced dataset of 4792 records was selected and the train/test split of 3300/1492 was chosen. Since the dataset contained 561 dimensions, the features were reduced to the closest square number (529) using information gain based feature selection. Then, each record was re-arranged into a 23×23 image.

5.5.2 Hyper-Parameter and Model Selection

As mentioned, we hypothesize that due to the parallel architecture of the presented DSOM, a shallower model compared to the original proposed deep SOM can be used to achieve the same pattern recognition capability. This results in a reduction of serial operations, resulting in reduced training time. In order to test this, for all the tests, a the original deep SOM with two hidden layers and an the novel DSOM with only one hidden layer were implemented. In the DSOM hidden layer, two parallel layers were implemented.

5.5.3 Experimental Results: Pattern Recognition Accuracy

In this section, to quantify the capability of pattern recognition, the labels of the test datasets were used. Each neuron of the output layer of DSOM was assigned a class label based on the frequency which it was picked as the winning neuron for each class. The class with the highest frequency was assigned as the label of the output neuron. Then, the classification accuracy scores were calculated for the three datasets. In addition, the generalization capability was analyzed by adding noise to the test data, and observing the change in classification accuracy. The presented

DSOM was compared to the originally proposed DSOM in [117]. All accuracy scores are given in 14. It should be noted that these results were published in [64].

MNIST

The original deep SOM was able to achieve the best test accuracy of 83.468% while the presented DSOM was able to achieve 87.118 % (A 3.65% improvement).

In terms of generalization capability, there was no significant difference in classification accuracy for both models until the noise level increased beyond 20%. Despite the drop in accuracy beyond 20% noise, it was observed that the presented DSOM consistently outperformed the original DSOM. Further, the presented method showed a lower generalization error at all the noise levels.

GSAD

DSOM achieved 57.24% as its best classification accuracy while presented DSOM achieved 72.73% (A 15.49% improvement).

Generalization capability: It was observed that presented DSOM outperformed DSOM at all noise levels. Further, the presented DSOM showed a lower generalization error at noise levels of 0%-20%.

SP-HAR

Original DSOM was able to achieve a maximum test accuracy of 57.88% while presented DSOM was able to achieve 64.36 (6.48% improvement).

In terms of generalization capability, presented DSOM outperformed the original DSOM at all noise levels except at 40% and 60% . Further, presented DSOM showed a lower generalization error for 0%-10% noise levels.

Table 14. Classification Accuracy comparison between the original DSOM and the DSOM presented in this dissertation. This establishes the pattern recognition capability of the presented methodology

Dataset	**Model**	**Test Acc.**	**Test Acc. for Noise Level** (%)						
			2	5	10	20	40	50	60
MNIST	Original	83.47	83.37	83.14	83.14	82.39	74.46	62.0	20.37
	Presented	87.12	87.12	87.15	86.88	86.51	79.91	69.34	23.63
GSAD	Original	57.24	49.76	45.20	38.08	32.59	27.12	23.84	21.88
	Presented	72.73	66.82	61.19	50.01	37.86	28.59	24.12	22.45
SP-HAR	Original	57.88	56.90	55.60	52.58	44.81	27.17	19.52	17.78
	Presented	64.36	63.22	61.90	58.22	48.51	24.14	19.69	17.35

5.6 Experimental Details of Explainabile DSOMs

This section presents the experiments conducted to test the explanation presented explanation methodology for the DSOM. This section uses the KDD-NSL dataset used in Chapter 4. Similarly to experiments conducted in Chapter 4, only data from Normal communication, Denial-of-Service and Probe were used in the testing. Several test cases were carried out to examine the explanation methodology. 1) Using all data to train the DSOM 2) Using only normal data and DoS to train the DSOM, 3) Using only normal data and probe data to train the DSOM. As mentioned, once the DSOM is trained, the output layer SOM is subjected to the explanation process to discover the learned behavior patterns.

Test case 1: Using all the data

In this test case, all the data from three classes were used to train the DSOM. Then, the output layer SOM was clustered using K-Means. Then, each cluster was

explained using the presented methodology.

Figures 23 and 25 shows the cluster label map and the feature behavior across the map for clustering the data into two clusters. This choice was made to have an unsupervised analysis akin to binary classification in Chapter 4. Upon examining Figure 25 it can be noticed that several features are behaving similarly and correlating with the cluster map. Tables 5.6 and 5.6 shows the top 20 Yager summaries generated for the clusters '0' and '1' respectively. It can be seen that, two clusters don't accurately represent the pattern in the data appropriately. The top summaries, while true, are summaries that indicate general behavior across the SOM. Therefore, the number of clusters that the SOM is clustered is crucial to ensure useful explanation. This can be viewed as a hyper-parameter that needs to be optimized through a form of cross-validation.

Figure 24 shows the cluster distribution when clustered three clusters. It can be observed that the cluster map correlates more with certain features when the granularity is increased. Tables 5.6 and 5.6 show the top 20 yager summaries in cluster '1' and '2' respectively. It can be seen that the yager summaries corresponding to high error rates appear in cluster '2'. Furthermore, it can be seen that certain features show the same behavior across multiple clusters. This means that the features are not contributing heavily to differentiate between the clusters. Therefore, those features can be filtered out when the final cluster descriptions are generated.

Test case 2: Using *Normal* and *DoS* data

In this test, the Probe data were not used to train the DSOM. Figures 26 and 27 show the cluster distribution across the DSOM and the feature behavior across the DSOM. It can be seen that features such as *same_srv_rate* show a correlation to the cluster map. Note that *same_srv_rate* was a feature that was highly influential for detecting *DoS* attacks in Chapter 4.

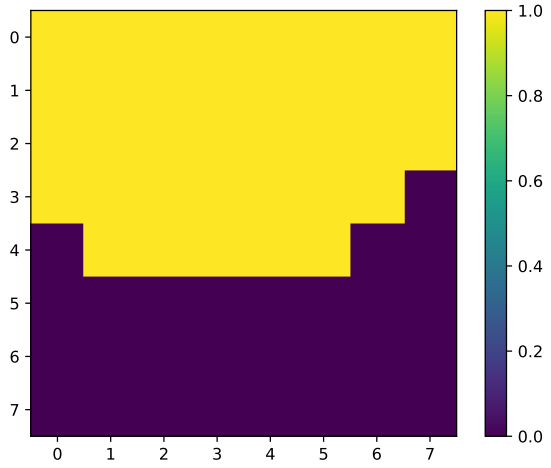


Fig. 23. Cluster distribution of the output SOM of the DSOM trained with all data. Weights of the SOM clustered to *two* classes using K-Means

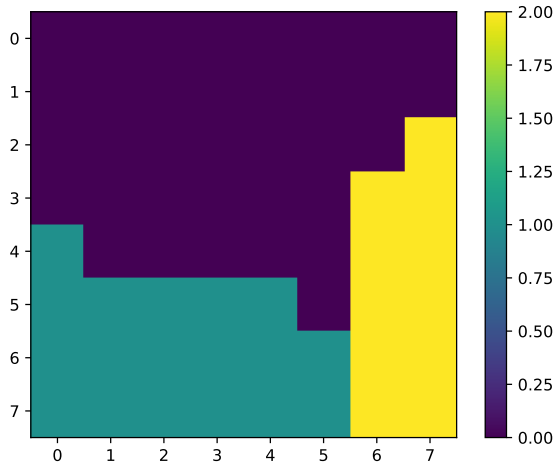


Fig. 24. Cluster distribution of the output SOM of the DSOM trained with all data. Weights of the SOM clustered to *three* classes using K-Means

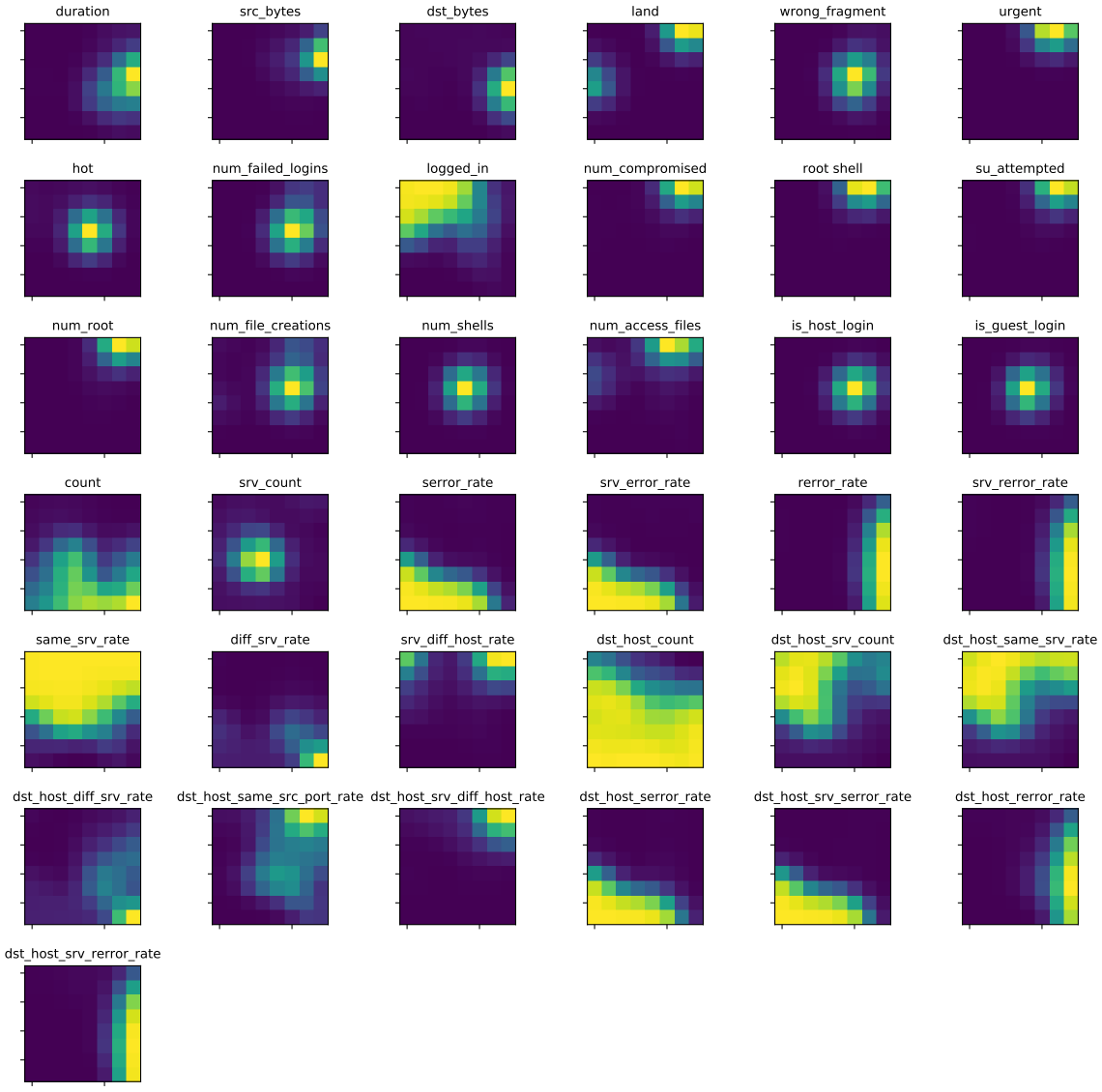


Fig. 25. Distribution of feature values across the DSOM. Trained using all data

Table 15. All data KDD-NSL, two-class clustering Yager Linguistic Summaries for cluster label '0'

Quantifier	**Feature**	**Feature_Value**	**Truth**
almost_all	land	low	1
almost_all	urgent	low	1
almost_all	hot	low	1
almost_all	num_compromised	low	1
almost_all	root shell	low	1
almost_all	su_attempted	low	1
almost_all	num_root	low	1
almost_all	num_shells	low	1
almost_all	num_access_files	low	1
almost_all	is_guest_login	low	1
almost_all	srv_diff_host_rate	low	1
almost_all	dst_host_srv_diff_host_rate	low	1
almost_all	is_host_login	low	0.999958
almost_all	num_file_creations	low	0.999958
almost_all	num_failed_logins	low	0.999874
almost_all	dst_host_same_srv_rate	low	0.999786
almost_all	dst_bytes	low	0.999714
almost_all	logged_in	low	0.998906
almost_all	srv_count	low	0.998904
almost_all	dst_host_srv_count	low	0.991329

Table 16. All data KDD-NSL, clustering with *two* clusters Yager Linguistic Summaries for cluster label '1'

Quantifier	**Feature**	**Feature_Value**	**truth**
almost_all	diff_srv_rate	low	1
almost_all	dst_host_srv_serror_rate	low	0.999933
almost_all	srv_error_rate	low	0.999908
almost_all	serror_rate	low	0.999907
almost_all	dst_host_serror_rate	low	0.999906
almost_all	src_bytes	low	0.999429
almost_all	dst_bytes	low	0.999316
almost_all	dst_host_srv_rerror_rate	low	0.998422
almost_all	dst_host_rerror_rate	low	0.997876
almost_all	srv_rerror_rate	low	0.997732
almost_all	rerror_rate	low	0.997448
almost_all	srv_count	low	0.994397
almost_all	same_srv_rate	high	0.993734
almost_all	wrong_fragment	low	0.99222
almost_all	is_host_login	low	0.987672
almost_all	num_compromised	low	0.986643
almost_all	dst_host_srv_diff_host_rate	low	0.986028
almost_all	num_access_files	low	0.985798
almost_all	num_failed_logins	low	0.985685
almost_all	num_root	low	0.98564

Table 17. All data KDD-NSL, clustering with *three* clusters, Yager Linguistic Summaries for cluster label '1'

Quantifier	**Feature**	**Feature_Value**	**truth**
almost_all	duration	low	1
almost_all	src_bytes	low	1
almost_all	dst_bytes	low	1
almost_all	land	low	1
almost_all	wrong_fragment	low	1
almost_all	urgent	low	1
almost_all	hot	low	1
almost_all	num_failed_logins	low	1
almost_all	num_compromised	low	1
almost_all	root shell	low	1
almost_all	su_attempted	low	1
almost_all	num_root	low	1
almost_all	num_file_creations	low	1
almost_all	num_shells	low	1
almost_all	num_access_files	low	1
almost_all	is_host_login	low	1
almost_all	is_guest_login	low	1
almost_all	rerror_rate	low	1
almost_all	srv_rerror_rate	low	1
almost_all	srv_diff_host_rate	low	1

Table 18. All data KDD-NSL, clustering with *three* clusters, Yager Linguistic Summaries for cluster label '2'

Quantifier	**Feature**	**Feature_Value**	**truth**
almost_all	land	low	1
almost_all	urgent	low	1
almost_all	hot	low	1
almost_all	logged_in	low	1
almost_all	num_compromised	low	1
almost_all	root shell	low	1
almost_all	su_attempted	low	1
almost_all	num_root	low	1
almost_all	num_shells	low	1
almost_all	num_access_files	low	1
almost_all	is_guest_login	low	1
almost_all	srv_count	low	1
almost_all	srv_diff_host_rate	low	1
almost_all	dst_host_same_srv_rate	low	1
almost_all	is_host_login	low	0.999854
almost_all	dst_host_count	high	0.987903
almost_all	srv_rerror_rate	high	0.986946
almost_all	rerror_rate	high	0.986625
almost_all	dst_host_srv_rerror_rate	high	0.986511
almost_all	dst_host_rerror_rate	high	0.985315

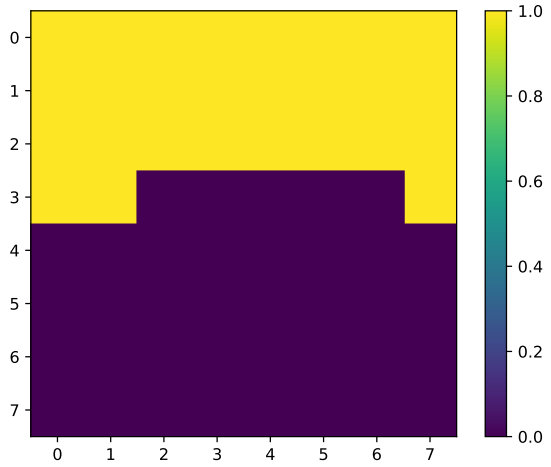


Fig. 26. Test Case 2: Cluster distribution of the output SOM of the DSOM trained with *Normal* and *DoS* data. Weights of the SOM clustered to *two* classes using K-Means

Tables 5.6 and 5.6 show the top 20 Yager summaries for the two clusters.

Test Case 3: Using *Normal* and *Probe* data

In this test case, *DoS* were not used in training the DSOM. The goal was to identify the behavior differences between *Normal* and *Probe* Figures 28 and 29 show the cluster distribution and feature behavior.

5.7 Evaluating the Features of the Explanation Methodology

Similar to the previous chapter, the presented methodology's features were evaluated against the requirements presented in Chapter 3. Table 23 summarizes the features of the presented unsupervised explainable CP-ADS methodology.

In terms of the types of explanations, the presented methodology only generates overall model explanations. A novel methodology needs to be developed to explain

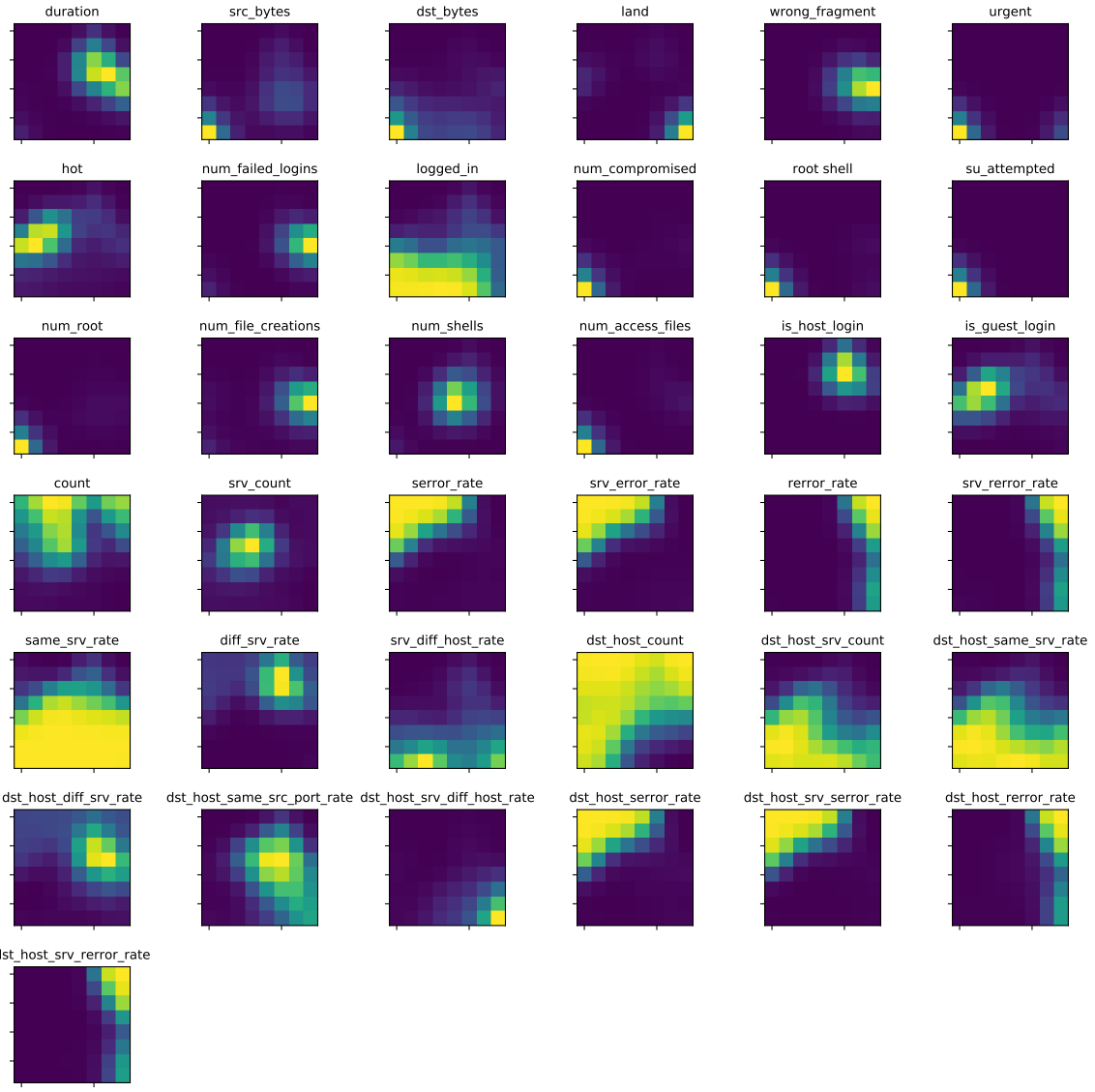


Fig. 27. Test Case 2: Distribution of feature values across the DSOM. Trained with *Normal* and *DoS* data

Table 19. *DoS* and *Normal* data KDD-NSL, clustering with *two* clusters, Yager Linguistic Summaries for cluster label '0'

Quantifier	**Feature**	**Feature_Value**	**truth**
almost_all	dst_host_srv_serror_rate	low	0.999498
almost_all	same_srv_rate	high	0.998967
almost_all	srv_error_rate	low	0.997574
almost_all	dst_host_srv_rerror_rate	low	0.997478
almost_all	serror_rate	low	0.997452
almost_all	dst_host_diff_srv_rate	low	0.996753
almost_all	is_guest_login	low	0.996403
almost_all	dst_host_serror_rate	low	0.996369
almost_all	dst_host_rerror_rate	low	0.996007
almost_all	duration	low	0.995521
almost_all	num_file_creations	low	0.99542
almost_all	is_host_login	low	0.995073
almost_all	dst_host_srv_diff_host_rate	low	0.994933
almost_all	srv_rerror_rate	low	0.994551
almost_all	rerror_rate	low	0.994431
almost_all	num_failed_logins	low	0.99123
almost_all	land	low	0.990816
almost_all	urgent	low	0.990002
almost_all	root shell	low	0.990002
almost_all	su_attempted	low	0.990002

Table 20. *DoS* and *Normal* data KDD-NSL, clustering with *two* clusters, Yager Linguistic Summaries for cluster label '1'

Quantifier	**Feature**	**Feature_Value**	**truth**
almost_all	src_bytes	low	1
almost_all	dst_bytes	low	1
almost_all	land	low	1
almost_all	wrong_fragment	low	1
almost_all	urgent	low	1
almost_all	num_failed_logins	low	1
almost_all	logged_in	low	1
almost_all	num_compromised	low	1
almost_all	root shell	low	1
almost_all	su_attempted	low	1
almost_all	num_root	low	1
almost_all	num_file_creations	low	1
almost_all	num_access_files	low	1
almost_all	srv_diff_host_rate	low	1
almost_all	dst_host_count	high	1
almost_all	dst_host_srv_diff_host_rate	low	1
almost_all	dst_host_srv_count	low	0.999691
almost_all	num_shells	low	0.999602
almost_all	dst_host_same_srv_rate	low	0.999346
almost_all	srv_count	low	0.997363

Table 21. *Probe* and *Normal* data KDD-NSL, clustering with *two* clusters, Yager Linguistic Summaries for cluster label '1'

Quantifier	**Feature**	**Feature_Value**	**truth**
almost_all	wrong_fragment	low	1
almost_all	urgent	low	1
almost_all	hot	low	1
almost_all	num_failed_logins	low	1
almost_all	num_compromised	low	1
almost_all	root shell	low	1
almost_all	su_attempted	low	1
almost_all	num_root	low	1
almost_all	num_shells	low	1
almost_all	num_access_files	low	1
almost_all	is_host_login	low	1
almost_all	is_guest_login	low	1
almost_all	srv_count	low	1
almost_all	dst_host_srv_serror_rate	low	0.998945
almost_all	dst_host_serror_rate	low	0.998677
almost_all	land	low	0.976562
almost_all	num_file_creations	low	0.974245
almost_all	srv_error_rate	low	0.974107
almost_all	logged_in	low	0.972524
almost_all	serror_rate	low	0.969225

Table 22. *Probe* and *Normal* data KDD-NSL, clustering with *two* clusters, Yager Linguistic Summaries for cluster label '0'

Quantifier	**Feature**	**Feature_Value**	**truth**
almost_all	duration	low	1
almost_all	src_bytes	low	1
almost_all	dst_bytes	low	1
almost_all	land	low	1
almost_all	wrong_fragment	low	1
almost_all	dst_host_srv_serror_rate	low	1
almost_all	num_file_creations	low	0.999881
almost_all	diff_srv_rate	low	0.999879
almost_all	hot	low	0.999817
almost_all	is_guest_login	low	0.999331
almost_all	num_failed_logins	low	0.999154
almost_all	su_attempted	low	0.999154
almost_all	is_host_login	low	0.999154
almost_all	num_access_files	low	0.999143
almost_all	urgent	low	0.999129
almost_all	dst_host_rerror_rate	low	0.999105
almost_all	num_compromised	low	0.999097
almost_all	dst_host_srv_diff_host_rate	low	0.999042
almost_all	srv_count	low	0.998957
almost_all	num_root	low	0.998889

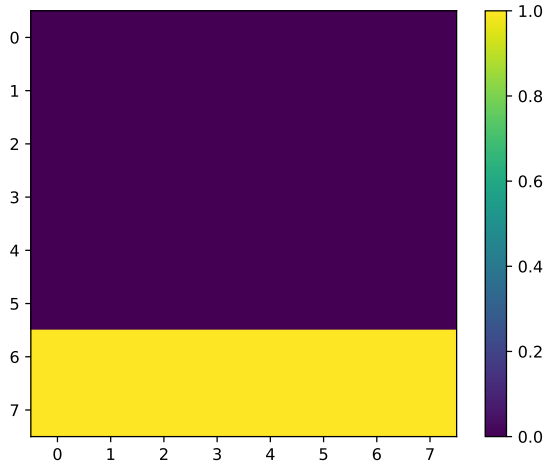


Fig. 28. Test Case 3: Cluster distribution of the output SOM of the DSOM trained with *Normal* and *Probe* data. Weights of the SOM clustered to *two* classes using K-Means

individual clustering decisions made by the presented algorithm. This is a future research area proposed in this work. As with the supervised case, the explanations of individual predictions should be easily and quickly understood given the time sensitivity.

In terms of human friendliness, the presented methodology is capable of generating explanations that are connected to the real-system behavior and helps the user understand the behavioral patterns of the CPS. In terms of completeness, the user can control the number of summaries are used to describe the CP behavior. However, more rigorous method of defining completeness should be defined and this requires further research. Furthermore, currently generated explanations are not contrastive and requires further research for generating contrastive explanations for unsupervised learning.

The explanations are generated in textual and visual form and thus, is commu-

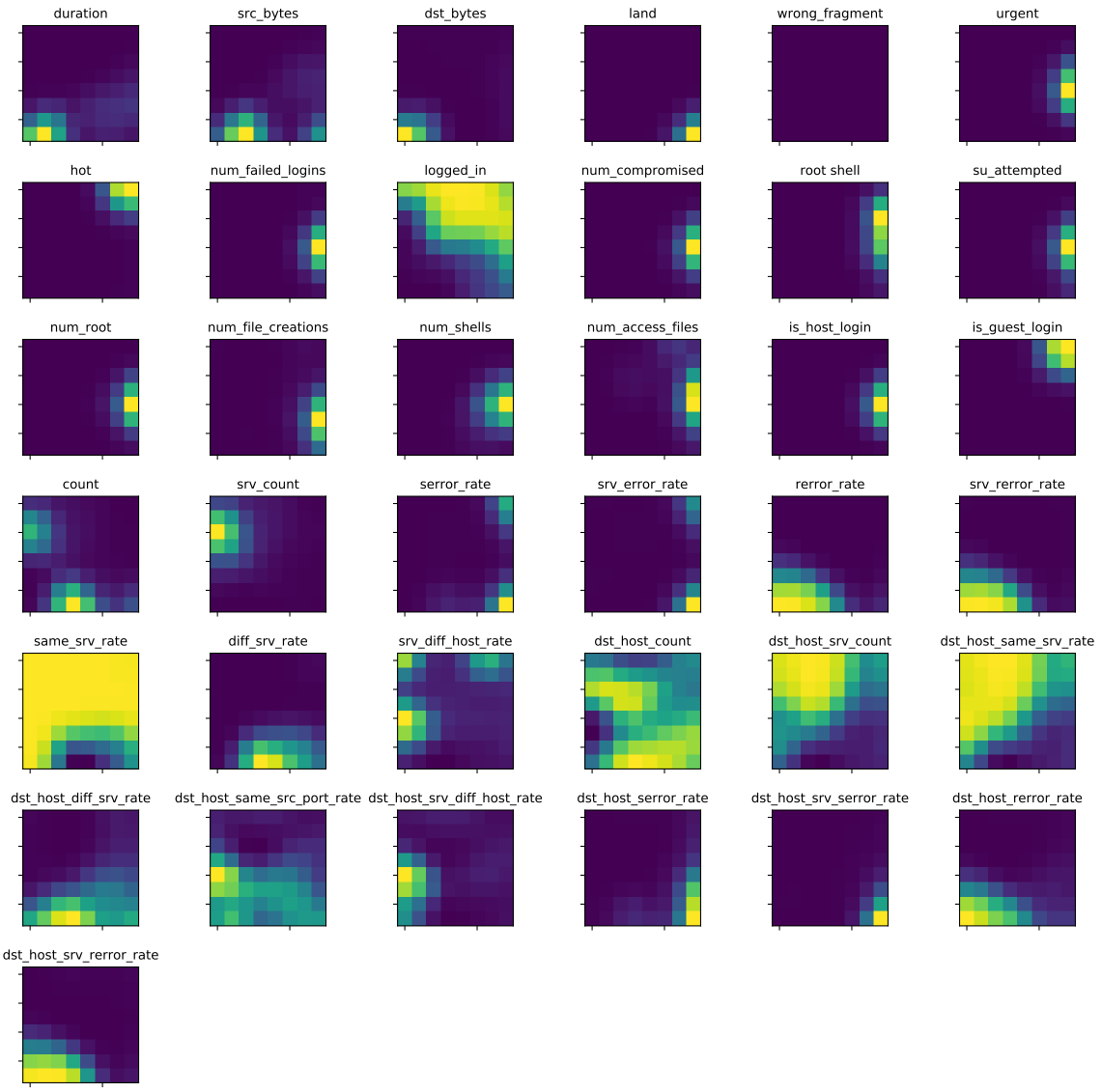


Fig. 29. Test Case 3: Distribution of feature values across the DSOM. Trained with *Normal* and *Probe* data

Table 23. Evaluating the presented unsupervised explainable CP-ADS methodology against the identified key-requirements

Requirements		**This Methodology**
Type of Explanation	Overall Model	Successfully generated
	Individual Predictions	Future work
Human-friendliness	Real-system behavior	Reflects the real-system
	Contrastive	Future work
	interpretability vs. completeness	Partially achieved
Explanation medium	Textual or visual	Textual and visual
Evaluation	Quantitative	Partially achieved with quality measures
	Qualitative	Partially achieved with visual validation

nicated in an understandable medium. In terms of evaluation, the presented quality measures are the only quantitative evaluation method presented in this dissertation and it is necessary to define a more rigorous one. The adversarial example generation methodology presented in Chapter 4 could be adapted to the unsupervised case, to generate perturbed examples to 'trick' the trained DSOM. The generated visualizations, the cluster map and the feature behavior heat-maps, can be viewed as a qualitative evaluation of the linguistic summaries. However, a more rigorous strategy involving end-users should be defined. Therefore, both evaluation requirements are only partially fulfilled.

5.8 Conclusions and Future work

This chapter presented a novel unsupervised neural network algorithm that can identify different behavioral patterns of a CPS. Unsupervised algorithms are used in tandem with supervised algorithms in CP-ADS to help identify previously-unseen intrusions/anomalies. The presented methodology is a multi-layered (deep) version of

Kohonen's Self-Organizing Map (DSOM). The presented DSOM algorithm retains all the advantages of SOMs and alleviates the limited feature abstraction capability of single-layer SOMs. The novel DSOM algorithm was implemented on several datasets and compared against other unsupervised neural networks. Experimental results verified the pattern recognition capability of DSOM. However, it was observed that the presented learning algorithm didn't support explainability. Therefore, the learning algorithm modified to preserve the dimensionality of the data throughout the network. Experimental results showed that the modification enabled deriving explanations of the trained DSOM and explaining the learned CP behavior of a CPS. It should be noted that the presented methodology does not support individual prediction explanation due to the noncontinuous transfer functions in the hidden layers.

The features of the presented methodology were evaluated against the requirements of explainability identified in Chapter 3 and future research needs were identified. In terms of the unsupervised learning algorithm, it will be further modified to enable continuous propagation of inputs throughout the network, enabling individual prediction explanations. In terms of explanations, future research would consider generating contrastive explanations and designing rigorous evaluation strategies, both quantitative and qualitative.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This chapter presents the overall conclusions of the presented work and proposes directions for future work.

6.1 Final Conclusions

The main objective of this dissertation was developing explainable, ANN-based CP-ADS methodologies, with the corollary of helping human operators trust ANN-based systems. The main objective was broken down into three sub-objectives: 1) identification of key requirements an explainable CP-ADSs should satisfy, 2) development of explainable supervised CP-ADS, and 3) development of explainable unsupervised CP-ADS.

Identification of explainability requirements: This dissertation argued that identifying context/domain-based desired features/requirements of explainability is an essential first step in the research. This step is often overlooked in existing explainable machine learning research. Furthermore, this dissertation argued that it is not possible to define domain-agnostic requirements for explainability since the explainability requirements are domain specific. Therefore, a set of requirements was discussed taking into account the unique properties of the domain. The requirements were discussed in terms of four factors: 1) what is explained, 2) when it is explained, 3) to whom it is explained, and 4) how it is explained. The identified requirements, though necessary, are not sufficient. To specify sufficient requirements, the end-users have to play a crucial part, and a collaboration between machine learning and social

science research is important. However, the presented set of requirements provides a base that can be expanded and refined. Further, the presented set of requirements lay the groundwork for building a framework to evaluate explainable algorithms.

This dissertation considered two cases of CP-ADSs: 1) supervised and 2) unsupervised, i.e. ANN-based classification and clustering. It was important to consider both cases for two reasons: 1) it is common to use both types of CP-ADSs in tandem to detect anomalies, and 2) from an explainability perspective, the approach and goals of explanation are different for supervised and unsupervised algorithms.

Explaining Supervised CP-ADS: This dissertation presented a methodology for deriving summaries of what the ANN classifier has learned about each anomaly type. The presented explanation methodology was successful in generating overall model explanations for the ANN classifier. Further, this dissertation presented a methodology for quantitatively validating the derived summaries. The validation methodology empirically validated the generated summaries. The presented methodology was evaluated against the set of explainability requirements identified in this dissertation. The methodology satisfied 5 out the 8 requirements. The dissertation proposed steps toward satisfying the other three–namely, individual prediction explanation, contrastive explanation generation and qualitative evaluation of the explanations. Steps were proposed to generate contrastive explanations and perform qualitative evaluations.

The summaries generated in this work help end-users gain insight into what the ANN-based CP-ADS has learned, and how it detects anomalies. This enables qualitative evaluation of the ANN-based CP-ADS. Before generating explanations, the evaluation was purely quantitative with accuracy scores. Accuracy scores enable answering the question "Is the CP-ADS doing the right thing?". Explanations make it possible to answer the question "Is the CP-ADS doing the right thing for the right

reasons?".

Explaining Unsupervised CP-ADS: This dissertation presented a novel ANN clustering methodology. It was designed to embed hierarchical feature abstraction capabilities to the single-layer Self-Organizing Maps (SOMs). The presented usupervised ANN, named Deep Self-Organizing Maps (DSOMs), was trained to learn Cyber-Physical behavioral patterns in a CPS using unlabeled data. The pattern recognition capability of the DSOM was established empirically through extensive experimentation. In addition to the DSOM, this dissertation presented a methodology for explaining the learned Cyber-Physical behavior to the user. The explanations summarized the feature behaviors (real-system behavior) that were dominant in the identified clusters. As with the supervised case, the presented methodology was evaluated against the explainability requirements from Chapter 3. The presented methodology satisfied three requirements and partially satisfies another three. The user's ability to adjust the interpretability vs completeness, and the evaluation of explanations—both, quantitative and qualitative—should be more rigorously defined and augmented. Further, the current methodology does not satisfy two requirements– namely, individual prediction explanation and contrastive explanation generation.

The explanations generated in this work enable an end-user to understand the different CP behaviors a system goes through and understand how each input feature (typically a system component) behave in that system operational state. Since visualizations and linguistic explanations are generated together, the user has the capability of validating the linguistic summaries visually.

Developing explainable CP-ADS systems is a process that requires research efforts from different areas to converge. This dissertation contributed with foundational methodologies for explaining supervised and unsupervised CP-ADS and laid out the foundation for an evaluation strategy by identifying explainability-requirements in

the domain of CP-ADS. Therefore, the contributions of this dissertation serve as a framework that can be expanded to further improve explainable ANN-based methods for CP-ADS. Furthermore, it should be noted that the overall approach of identifying requirements of explainability around the specified four criteria can be transferred to other problem domains that require explainable machine learning models.

6.2 Future Research Directions

There are several possible future research directions for the presented work. This section attempts to enumerate several immediate next steps to further develop the presented work.

In terms of generated explanations, one important improvement is deriving contrastive explanations. The derived explanations provide insight into the factors that influence CP-ADS's decision toward a certain intrusion type. It doesn't provide insight into what factors influence the decision *toward the intrusion type over another class*. Lipton pointed out that humans tend to reason using not just *"why this?"* but also *"why not something else?"* [129]. Therefore, the explanation methodology presented in Chapter 4 needs to be extended to support this. First, the negative relevance scores that were ignored in the work can be used to summarize the feature behavior that *negatively* influences the detection of a certain intrusion type. These summaries can be combined with the *positive* influence summaries to generate contrastive explanations. Second, the IF-THEN summary evaluation methodology using adversarial examples presented in Chapter 4 can be extended to generated counterfactual explanations. Counterfactual explanations answer the question "how would the predictions change if feature behavior was different?" [130]. In the adversarial examples based method, the feature behavior was changed to empirically prove that the derived explanations are correct. This principle can be extended to strengthen

the contrastive explanations with counterfactual explanations.

The unsupervised learning algorithm presented in Chapter 5 (DSOM) employs winner-take-all (WTA) competitive learning to learn the weights of the self-organizing neurons. While this helps organize the neuron grid to mimic the topological properties of the data, the current method of picking the "winning neuron" or the best-matching-unit, doesn't allow a continuous propagation of the input features through the multiple layers. If a continuous propagation could be achieved, gradient-based optimization methodologies could be utilized in conjunction with WTA to potentially improve the learning capabilities of the DSOM. There have been several attempts to combine WTA with gradient descent methodologies with Kohonen's SOMs [131], [132]. These studies could be extended to the DSOM algorithm to improve it further.

In the current explanations methodology presented in Chapter 5, it is possible to generate only model-level explanations that are capable of summarizing the clusters. With the current methodology, it is not possible to observe how each input feature contributes to the clustering decisions through the layers. As mentioned above, the current WTA methodology propagates the data with non-continuous transfer functions preventing the back-propagation of relevance similar to the LRP method used in Chapter 4. If the learning algorithm is modified as mentioned above, a new relevance propagation method could be defined to identify how each feature contributes to each clustering decision.

The methodologies presented in this dissertation focused on deriving summaries of the overall knowledge of neural networks. Further, the explanations were delivered to the user in a linguistic format. However, not much was discussed about explanations of individual predictions made by the CP-ADS. While there are several saliency-mapping techniques available in the literature (for the supervised case), this is a non-trivial task as responding to an alert from the CP-ADS is an extremely

time-sensitive matter. Hence, it requires explanations that could be easily and quickly grasped. Therefore, devising strategies to optimally present explanations of individual predictions is another multi-disciplinary research direction. If successful, it could help end-users trust the predictions made by CP-ADS and make more informed actions. The intersection of human-factors and machine learning could be explored to develop methodologies to quickly convey the most important reasons behind a prediction.

Appendix A

LIST OF PUBLICATIONS BY THE AUTHOR

This appendix presents a list of the author's published and submitted peer-reviewed publications.

Peer-reviewed Journals:

1. K. Amarasinghe, M. Manic, "What does it Mean to be Explainable? Explainable Artificial Intelligence for Safety-Critical Systems" in review for IEEE Industrial Electronics Magazine.

2. K. Amarasinghe, M. Manic, "Explainable Neural Networks based Intrusion Detection Systems" in review for IEEE Transactions on Industrial Informatics.

3. C. S. Wickramasinghe, K. Amarasinghe, M. Manic, "Deep Self-Organizing Maps for Unsupervised Image Classification" in IEEE Transactions on Industrial Informatics 2019 (Early Access).

4. R. Fernandez Molanes, K. Amarasinghe, J. Rodriguez-Andina, and M. Manic, "Deep Learning and Reconfigurable Platforms in the Internet of Things: Challenges and Opportunities in Algorithms and Hardware," in IEEE Industrial Electronics Magazine, vol. 12, no. 2, pp. 36-49, June 2018.

5. M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, C. Rieger, " Buildings of the future: energy efficient and powered by artificial intelligence, connected and cyber aware, and human-driven," in IEEE Industrial Electronics Magazine, vol. 10, no. 4, pp. 32-49, Dec. 2016.

6. M. Manic, D. Wijayasekara, K. Amarasinghe, and J. J. Rodriguez-Andina, "Building Energy Management Systems: The Age of Intelligent and Adaptive Buildings," in IEEE Industrial Electronics Magazine, vol. 10, no. 1, pp. 25-39, Spring 2016.

Book Chapters:

7. P. Sivils, C. Rieger, K. Amarasinghe, M. Manic, Integrated Cyber-Physical Assessment and Response for Improved Resiliency, chapter in The Internet of Things for Smart Urban Ecosystems, Springer, Jan 2019.

Peer-reviewed Conference Proceedings:

8. K. Amarasinghe, M. Manic, "Explaining What a Neural Network has Learned: Toward Transparent Classification" in Proc. Advances on eXplainable AI, International Conference on Fuzzy Systems, FUZZ-IEEE 2019, New Orleans, LA, USA, Jun. 23-26, 2019

9. K. Amarasinghe, M. Manic, "Improving User Trust on Deep Neural Networks based Intrusion Detection Systems" in Proc. 44rd Annual Conference of the IEEE Industrial Electronics Society, IECON 2018, Washington DC, USA, Oct. 21-23, 2018

10. K. Amarasinghe, C. Wickramasinghe, D. Marino, C.Rieger, M. Manic, "Framework for Data-Driven Health Monitoring of Cyber-Physical Systems", IEEE Resilience Week (RW) 2018, Denver, CO, USA, Aug 20-23, 2018.

11. K. Amarasinghe, K. Kenney, and M. Manic, "Toward Explainable Deep Neural Network based Anomaly Detection", in Proc. 11th International Conference on Human System Interaction, IEEE HSI 2018, Gdansk, Poland, July 04-06, 2018.

12. C. Wickramasinghe, K. Amarasinghe, D. Marino, and M. Manic, Deep Self-Organizing Maps for Visual Data Mining, in Proc. 11th International Conference on Human System Interaction, IEEE HSI 2018, Gdansk, Poland, July 04-06, 2018.

13. C. Wickramasinghe, K. Amarasinghe, M. Manic, "Parallelizable Deep Self-Organizing Maps for Image Classification", in Proc. 2017 IEEE Symposium Series on Computational Intelligence, IEEE SSCI 2017, Honolulu, Hawaii, USA, Nov, 27- Dec 1, 2017.

14. D. Marino, K. Amarasinghe, M. Anderson, N. Yancey, Q. Nguyen, K. Kenney, M. Manic, "Data-driven decision support for reliable biomass feedstock preprocessing", IEEE Resilience Week (RWS) 2017, Wilmington, DE, USA, Sep 18-22, 2017.

15. H. J. Carey, K. Amarasinghe, M. Manic, "Reduction of Massive EEG Datasets for Epilepsy Analysis using Artificial Neural Networks" in Proc. 10th International Conference on Human System Interaction, IEEE HSI 2017, Ulsan, Republic of Korea, July 17-19, 2017.

16. K. Amarasinghe, D. Marino, M. Manic, "Deep learning for Energy Load Forecasting" in Press. in Proc. 26th International Symposium on Industrial Electronics, IEEE ISIE 2017, Edinburgh, Scotland, June 19-21, 2017.

17. D. Marino, K. Amarasinghe, M. Manic, "Simultaneous Generation-Classification Using LSTM," in Proc. 2016 IEEE Symposium Series on Computational Intelligence IEEE SSCI 2016, Athens, Greece, Dec. 6-9, 2016.

18. D. Marino, K. Amarasinghe, M. Manic, "Building Energy Load Forecasting using Deep Neural Networks," in Press. 42nd Annual Conference of the IEEE

Industrial Electronics Society, IEEE IECON 2016, Florence, Italy, Oct. 24-27, 2016.

19. K. Amarasinghe, D. Wijayasekara, H. Carey, M. Manic, D. He, W. Chen, "Artificial Neural Networks based Thermal Energy Storage Control for Buildings," in Proc. 41st Annual Conference of the IEEE Industrial Electronics Society, IEEE IECON 2015, Yokohama, Japan, Nov. 09-12, 2015.

20. K. Amarasinghe, M. Manic, R. Hruska, "Optimal Stop Word Selection for Text Mining in Critical Infrastructure Domain," in Proc. IEEE Symposium on Resilience Control Systems, ISRCS 2015, Philadelphia, PA, Aug. 18-20, 2015.

21. K. Amarasinghe, D. Wijayasekara, M. Manic, "EEG based brain activity monitoring using Artificial Neural Networks," in Proc. of IEEE 7th International Conference on Human System Interaction, IEEE HSI 2014, Lisbon, Portugal, June 16-18, 2014.

22. K. Amarasinghe, D. Wijayasekara, M. Manic, "Neural Network-Based Downscaling of Building Energy Management System Data," in Proc. IEEE International Symposium on Industrial Electronics, ISIE 2014, Istanbul, Turkey, June 1-4, 2014.

REFERENCES

[1] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[2] Apple. *Siri - Apple*. URL: https://www.apple.com/siri/ (visited on 11/28/2018).

[3] Google. *Google Assistant*. URL: https://assistant.google.com/platforms/phones/ (visited on 11/28/2018).

[4] BBC. *Will 5G be necessary for self-driving cars? - BBC News*. URL: https://www.bbc.com/news/business-45048264 (visited on 11/28/2018).

[5] The Verges. *Just how good is Google Word Lens at deciphering Japanese? - The Verge*. URL: https://www.theverge.com/2017/1/27/14409142/google-translate-japanese-word-lens-update (visited on 11/28/2018).

[6] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521 (2015), pp. 436–444.

[7] Jason Bloomberg. "Don't Trust Artificial Intelligence? Time To Open The AI 'Black Box'". In: (2018).

[8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You? Explaining the Predictions of Any Classifier". In: *KDD*. 2016.

[9] IBM. *What's next for AI Building trust*. URL: https://www.ibm.com/watson/advantage-reports/future-of-artificial-intelligence/building-trust-in-ai.html (visited on 11/28/2018).

[10] Sebastian Bach et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLOS ONE* 10.7 (2015), e0130140.

[11] Wojciech Samek et al. "Evaluating the Visualization of What a Deep Neural Network Has Learned". In: *IEEE Trans. Neural Networks Learn. Syst.* 28.11 (2017), pp. 2660–2673. ISSN: 2162-237X.

[12] David Gunning. *Explainable Artificial Intelligence*. 2016. URL: `https://www.darpa.mil/program/explainable-artificial-intelligence` (visited on 04/26/2018).

[13] Dumitru Erhan et al. "Visualizing Higher-Layer Features of a Deep Network". In: (2009).

[14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: (2013). arXiv: `1312.6034`.

[15] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. "Adaptive deconvolutional networks for mid and high level feature learning". In: *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2018–2025. ISBN: 978-1-4577-1102-2.

[16] Zachary C. Lipton. "The Mythos of Model Interpretability". In: (2016). arXiv: `1606.03490`.

[17] Lisa Anne Hendricks et al. "Generating Visual Explanations". In: (2016). arXiv: `1603.08507`.

[18] Supriyo Chakraborty et al. “Interpretability of Deep Learning Models: A Survey of Results”. In: *IEEE Smart World Congress : DAIS - Workshop on Distributed Analytics InfraStructure and Algorithms for Multi-Organization Federations*. 2017.

[19] Sushant Jain et al. “Exploiting Mobility for Energy Efficient Data Collection in Wireless Sensor Networks”. In: *Mob. Networks Appl.* 11.3 (2006), pp. 327–339. ISSN: 1383-469X.

[20] Radhakisan Baheti and Helen Gill. “Cyber-Physical Systems”. In: *impact Control Technol.* 12.1 (2011), pp. 161–166.

[21] Jayavardhana Gubbi et al. “Internet of Things (IoT): A vision, architectural elements, and future directions”. In: *Futur. Gener. Comput. Syst.* 29.7 (2013), pp. 1645–1660. ISSN: 0167-739X.

[22] Finale Doshi-Velez and Been Kim. “Towards A Rigorous Science of Interpretable Machine Learning”. In: (2017). arXiv: 1702.08608.

[23] Amina Adadi and Mohammed Berrada. “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6 (2018), pp. 52138–52160. ISSN: 21693536.

[24] Song Han et al. “Intrusion detection in cyber-physical systems: Techniques and challenges”. In: *IEEE Syst. J.* 8.4 (2014), pp. 1049–1059. ISSN: 19379234.

[25] Jonathan Goh et al. “Anomaly Detection in Cyber Physical Systems Using Recurrent Neural Networks”. In: *2017 IEEE 18th Int. Symp. High Assur. Syst. Eng.* (2017), pp. 140–145. ISSN: 15302059.

[26] P.J. Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560. ISSN: 00189219.

[27] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference for Learning Representations*. San Diego, 2015. arXiv: 1412.6980.

[28] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Methods for Interpreting and Understanding Deep Neural Networks". In: (2017). arXiv: 1706.07979.

[29] David Baehrens et al. "How to Explain Individual Classification Decisions". In: *Journal of Machine Learning Research* 11 (2010), pp. 1803–1831.

[30] Matthew D Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: (2014). arXiv: 1311.2901.

[31] Jeff Donahue et al. "Long-term Recurrent Convolutional Networks for Visual Recognition and Description". In: (2014). arXiv: 1411.4389.

[32] NIST. *Framework for Cyber-Physical Systems: Volume 1, Overview*. Tech. rep. 2017.

[33] Edward A. Lee. "CPS foundations". In: *Proceedings of the 47th Design Automation Conference on - DAC '10*. New York, New York, USA: ACM Press, 2010, p. 737. ISBN: 9781450300025.

[34] Roberto Fernandez Molanes et al. "Deep Learning and Reconfigurable Platforms in the Internet of Things: Challenges and Opportunities in Algorithms and Hardware". In: *IEEE Industrial Electronics Magazine* 12.2 (2018), pp. 36–49. ISSN: 1932-4529.

[35] Marija D. Ilic et al. "Modeling of Future CyberPhysical Energy Systems for Distributed Sensing and Control". In: *IEEE Transactions on Systems, Man,*

and Cybernetics - Part A: Systems and Humans 40.4 (2010), pp. 825–838. ISSN: 1083-4427.

[36] Wayne Wolf. "Cyber-physical systems". In: *IEEE Computer* 42.3 (2009), pp. 88–89. ISSN: 00189162.

[37] Ying Tan, Steve Goddard, and Lance C. Pérez. "A prototype architecture for cyber-physical systems". In: *ACM SIGBED Review* 5.1 (2008), pp. 1–2. ISSN: 15513688.

[38] Song Han et al. "Intrusion detection in cyber-physical systems: Techniques and challenges". In: *IEEE Systems Journal* 8.4 (2014), pp. 1049–1059. ISSN: 19379234.

[39] Thomas H. Morris et al. "Engineering future cyber-physical energy systems: Challenges, research needs, and roadmap". In: *41st North American Power Symposium, NAPS 2009*. 2009, pp. 1–6. ISBN: 9781424444281.

[40] Victoria J. Hodge and Jim Austin. "A Survey of Outlier Detection Methodologies". In: *Artificial Intelligence Review* 22.2 (2004), pp. 85–126. ISSN: 0269-2821.

[41] Vic. Barnett and Toby. Lewis. *Outliers in statistical data*. Wiley, 1994, p. 584. ISBN: 9780471930945.

[42] R. a. Kisner et al. *Cybersecurity through Real-Time Distributed Control Systems*. Tech. rep. February. 2010, pp. 1–28.

[43] Robert Mitchell and Ing-Ray Chen. "A survey of intrusion detection techniques for cyber-physical systems". In: *ACM Computing Surveys* 46.4 (2014), pp. 1–29. ISSN: 03600300.

[44] Fariba Haddadi and Mehdi A. Sarram. "Wireless Intrusion Detection System Using a Lightweight Agent". In: *2010 Second International Conference on Computer and Network Technology*. IEEE, 2010, pp. 84–87. ISBN: 978-1-4244-6961-1.

[45] Michael E. Whitman and Herbert J. Mattord. *Principles of information security*. Course Technology, 2012, p. 617. ISBN: 1111138214.

[46] Yunlu Gong et al. "Intrusion Detection System Combining Misuse Detection and Anomaly Detection Using Genetic Network Programming". In: *ICROS-SICE International Joint Conference 2009* (2009), pp. 3463–3467.

[47] Yang Xia Luo. "The research of Bayesian classifier algorithms in intrusion detection system". In: *Proceedings of the International Conference on E-Business and E-Government, ICEE 2010* (2010), pp. 2174–2178. ISSN: 1355-2554.

[48] Ondrej Linda, Todd Vollmer, and Milos Manic. "Neural Network based Intrusion Detection System for critical infrastructures". In: *2009 International Joint Conference on Neural Networks*. IEEE, 2009, pp. 1827–1834. ISBN: 978-1-4244-3548-7.

[49] Istvan Kiss et al. "Data clustering-based anomaly detection in industrial control systems". In: *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2014, pp. 275–281. ISBN: 978-1-4799-6569-4.

[50] Alfonso Valdes, Richard Macwan, and Matthew Backes. "Anomaly Detection in Electrical Substation Circuits via Unsupervised Machine Learning". In: *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*. IEEE, 2016, pp. 500–505. ISBN: 978-1-5090-3207-5.

[51] Po-Yu Chen, Shusen Yang, and Julie A. McCann. "Distributed Real-Time Anomaly Detection in Networked Industrial Sensing Systems". In: *IEEE Transactions on Industrial Electronics* 62.6 (2015), pp. 3832–3842. ISSN: 0278-0046.

[52] Chao Liu et al. "An Unsupervised Spatiotemporal Graphical Modeling Approach to Anomaly Detection in Distributed CPS". In: *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–10. ISBN: 978-1-5090-1772-0.

[53] Hamidreza Sadreazami et al. "Distributed-Graph-Based Statistical Approach for Intrusion Detection in Cyber-Physical Systems". In: *IEEE Transactions on Signal and Information Processing over Networks* 4.1 (2018), pp. 137–147. ISSN: 2373-776X.

[54] Anna Magdalena Kosek and Oliver Gehrke. "Ensemble regression model-based anomaly detection for cyber-physical intrusion detection in smart grids". In: *2016 IEEE Electrical Power and Energy Conference (EPEC)*. IEEE, 2016, pp. 1–7. ISBN: 978-1-5090-1919-9.

[55] Yoshiyuki Harada et al. "Log-Based Anomaly Detection of CPS Using a Statistical Method". In: *2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. IEEE, 2017, pp. 1–6. ISBN: 978-1-5090-6699-5.

[56] Dawei Shi et al. "Causality Countermeasures for Anomaly Detection in Cyber-Physical Systems". In: *IEEE Transactions on Automatic Control* 63.2 (2018), pp. 386–401. ISSN: 0018-9286.

[57] Vasundhara Gokarn, Vaishali Kulkarni, and Prateek Singh. "Enhancing cyber physical system security via anomaly detection using behaviour analysis". In: *2017 International Conference on Wireless Communications, Signal Process-*

ing and Networking (WiSPNET). IEEE, 2017, pp. 944–948. ISBN: 978-1-5090-4442-9.

[58] Hasan Yetis and Mehmet Karakose. "Image processing based anomaly detection approach for synchronous movements in cyber-physical systems". In: *2018 23rd International Scientific-Professional Conference on Information Technology (IT)*. IEEE, 2018, pp. 1–4. ISBN: 978-1-5386-3620-6.

[59] Tim Miller, Piers Howe, and Liz Sonenberg. "Explainable AI: Beware of Inmates Running the Asylum". In: (2017). arXiv: 1712.00547v2.

[60] Jichen Zhu et al. "Explainable AI for Designers: A Human-Centered Perspective on Mixed-Initiative Co-Creation". In: *IEEE Conf. Comput. Intell. Games, CIG*. Vol. 2018-Augus. IEEE Computer Society, 2018. ISBN: 9781538643594.

[61] Gheorghe Sebestyen and Anca Hangan. "Anomaly detection techniques in cyber-physical systems". In: *Acta Univ. Sapientiae, Inform.* 9.2 (2017), pp. 101–118. ISSN: 2066-7760.

[62] Paul Vincent S. Alpano, Jhoanna Rhodette I. Pedrasa, and Rowel Atienza. "Multilayer perceptron with binary weights and activations for intrusion detection of Cyber-Physical systems". In: *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON* 2017-Decem (2017), pp. 2825–2829. ISSN: 21593450.

[63] Basant Subba, Santosh Biswas, and Sushanta Karmakar. "A Neural Network based system for Intrusion Detection and attack classification". In: *2016 22nd Natl. Conf. Commun. NCC 2016*. Institute of Electrical and Electronics Engineers Inc., 2016. ISBN: 9781509023615.

[64] Chathurika S. Wickramasinghe, Kasun Amarasinghe, and Milos Manic. "Deep Self-Organizing Maps for Unsupervised Image Classification". In: *IEEE Trans. Ind. Informatics* (2019), pp. 1–1. ISSN: 1551-3203.

[65] Donghwoon Kwon et al. "A survey of deep learning-based network anomaly detection". In: *Cluster Comput.* (2017), pp. 1–13.

[66] Ribana Roscher et al. "Explainable Machine Learning for Scientific Insights and Discoveries". In: (2019). arXiv: 1905.08883v2.

[67] Aditya Ashok, Manimaran Govindarasu, and Jianhui Wang. "Cyber-Physical Attack-Resilient Wide-Area Monitoring, Protection, and Control for the Power Grid". In: *Proc. IEEE* 105.7 (2017), pp. 1389–1407. ISSN: 00189219.

[68] Tim Miller. "Explanation in Artificial Intelligence: Insights from the Social Sciences". In: (). arXiv: 1706.07269v3.

[69] Leilani H Gilpin et al. *Explaining Explanations: An Overview of Interpretability of Machine Learning*. Tech. rep. 2019. arXiv: 1806.00069v3.

[70] Pauline Dewan. "Words Versus Pictures: Leveraging the Research on Visual Communication". In: *Partnersh. Can. J. Libr. Inf. Pract. Res.* 10.1 (2015), pp. 1–10. ISSN: 1911-9593.

[71] Christian Callegari, Stefano Giordano, and Michele Pagano. "Neural network based anomaly detection". In: *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2014, pp. 310–314. ISBN: 978-1-4799-5725-5.

[72] T. Brotherton and T. Johnson. "Anomaly detection for advanced military aircraft using neural networks". In: *2001 IEEE Aerospace Conference Pro-*

ceedings (Cat. No.01TH8542). Vol. 6. IEEE, 2001, pp. 3113–3123. ISBN: 0-7803-6599-2.

[73] Ronald R. Yager. "A new approach to the summarization of data". In: *Information Sciences* 28.1 (1982), pp. 69–86. ISSN: 0020-0255.

[74] G. Raschia and N. Mouaddib. "Using fuzzy labels as background knowledge for linguistic summarization of databases". In: *IEEE International Conference on Fuzzy Systems* 3 (2002), pp. 1372–1375.

[75] Dongrui Wu, Jerry M. Mendel, and Jhiin Joo. "Linguistic summarization using IF-THEN rules". In: *International Conference on Fuzzy Systems*. IEEE, 2010, pp. 1–8. ISBN: 978-1-4244-6919-2.

[76] Dongrui Wu and Jerry M. Mendel. "Linguistic summarization using IFTHEN rules and interval Type-2 fuzzy sets". In: *IEEE Transactions on Fuzzy Systems* 19.1 (2011), pp. 136–151. ISSN: 10636706.

[77] L.A. Zadeh. "Fuzzy sets". In: *Information and Control* 8.3 (1965), pp. 338–353. ISSN: 0019-9958.

[78] Kaoru Hirota and Witold Pedrycz. "Fuzzy computing for data mining". In: *Proceedings of the IEEE* 87.9 (1999), pp. 1575–1600. ISSN: 00189219.

[79] Kasun Amarasinghe and Milos Manic. "Explaining What a Neural Network has Learned: Toward Explainable Classification". In: *FUZZ-IEEE*. 2019.

[80] Christian Szegedy et al. "Intriguing properties of neural networks". In: (2013). arXiv: 1312.6199.

[81] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: (2014).

[82] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. "One pixel attack for fooling deep neural networks". In: (2017).

[83] Mahbod Tavallaee et al. "A detailed analysis of the KDD CUP 99 data set". In: *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009* Cisda (2009), pp. 1–6. ISSN: 2329-6267.

[84] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323. ISSN: 00189219.

[85] Xiang Zhang, Junbo Zhao, and Yann LeCun. *Character-level Convolutional Networks for Text Classification*. 2015.

[86] Friedhelm Schwenker and Edmondo Trentin. "Pattern classification and clustering: A review of partially supervised learning approaches". In: *Pattern Recognition Letters* 37 (2014), pp. 4–14. ISSN: 0167-8655.

[87] Di Wu et al. "A Highly Accurate Framework for Self-Labeled Semisupervised Classification in Industrial Applications". In: *IEEE Transactions on Industrial Informatics* 14.3 (2018), pp. 909–920. ISSN: 1551-3203.

[88] M.R.G. Meireles, P.E.M. Almeida, and M.G. Simoes. "A comprehensive review for industrial applicability of artificial neural networks". In: *IEEE Transactions on Industrial Electronics* 50.3 (2003), pp. 585–601. ISSN: 0278-0046.

[89] J.F. Martins, V.F. Pires, and A.J. Pires. "Unsupervised Neural-Network-Based Algorithm for an On-Line Diagnosis of Three-Phase Induction Motor Stator Fault". In: *IEEE Transactions on Industrial Electronics* 54.1 (2007), pp. 259–264. ISSN: 0278-0046.

[90] Ravi Kiran, Mathew Thomas, and Ranjith Parakkal. "An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos". In: (2018).

[91] Geoffrey E. Hinton et al. "The "wake-sleep" algorithm for unsupervised neural networks". In: *Science* 268.5214 (1995), pp. 1158–1161. ISSN: 00368075.

[92] Teuvo Kohonen. "The Self-organizing Map". In: *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[93] P. Lichodzijewski, A. Nur Zincir-Heywood, and M.I. Heywood. "Host-based intrusion detection using self-organizing maps". In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290).* IEEE, 2002, pp. 1714–1719. ISBN: 0-7803-7278-6.

[94] Yu Chia Hsu and An Pin Chen. "Clustering time series data by SOM for the optimal hedge ratio estimation". In: *Proceedings - 3rd International Conference on Convergence and Hybrid Information Technology, ICCIT 2008* 2 (2008), pp. 1164–1169.

[95] A. Rauber, D. Merkl, and M. Dittenbach. "The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data". In: *IEEE Transactions on Neural Networks* 13.6 (2002), pp. 1331–1341. ISSN: 1045-9227.

[96] C.S. Wickramasinghe, K. Amarasinghe, and M. Manic. "Parallalizable deep self-organizing maps for image classification". In: *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings*. Vol. 2018-Janua. 2017. ISBN: 9781538627259.

[97] C.S. Wickramasinghe et al. "Deep Self-Organizing Maps for Visual Data Mining". In: *11th International Conference on Human System Interaction (HSI)*. 2018, pp. 304–310.

[98] Teuvo Kohonen. "Self-organized formation of topologically correct feature maps". In: *Biological Cybernetics* 43.1 (1982), pp. 59–69. ISSN: 0340-1200.

[99] A. Ultsch. "Self-Organizing Neural Networks for Visualisation and Classification". In: Springer, Berlin, Heidelberg, 1993, pp. 307–313.

[100] Deboeck Guido, Teuvo Kohonen, and Carlos Cinca. *Visual Intelligence in Finance using Self-Organizing Maps*. 1997.

[101] Dumidu Wijayasekara and Milos Manic. "Visual, linguistic data mining using Self- Organizing Maps". In: *The 2012 International Joint Conference on Neural Networks (IJCNN)*. Brisbane, QLD, Australia: IEEE, 2012, pp. 1–8. ISBN: 978-1-4673-1490-9.

[102] G a Barreto and a R Araujo. "Identification and control of dynamical systems using the self-organizing map." In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 15.5 (2004), pp. 1244–1259. ISSN: 1045-9227.

[103] Teuvo Kohonen et al. "Engineering Applications of the Self-Organizing Map". In: *Proceedings of the IEEE* 84.10 (1996).

[104] Barbara Hammer et al. "Self-Organizing Maps for Time Series". In: *WSOM 5th Workshop On Self-Organizing Maps* (2005), pp. 1–8.

[105] M. Hagenbuchner, A. Sperduti, and Ah Chung Tsoi. "A self-organizing map for adaptive processing of structured data". In: *IEEE Transactions on Neural Networks* 14.3 (2003), pp. 491–505. ISSN: 1045-9227.

[106] Teuvo Kohonen. "Self-Organization of Very Large Document Collections: State of the Art". In: 1998, pp. 65–74.

[107] Michal Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 3844–3852. (Visited on 10/18/2017).

[108] D. Wu et al. "A Highly-Accurate Framework for Self-Labeled Semi-Supervised Classification in Industrial Applications". In: *IEEE Transactions on Industrial Informatics* PP.99 (2017), pp. 1–1. ISSN: 1551-3203.

[109] F. Luo et al. "Advanced Pattern Discovery-based Fuzzy Classification Method for Power System Dynamic Security Assessment". In: *IEEE Transactions on Industrial Informatics* 11.2 (Apr. 2015), pp. 416–426. ISSN: 1551-3203.

[110] M. Cococcioni, B. Lazzerini, and S. L. Volpi. "Robust Diagnosis of Rolling Element Bearings Based on Classification Techniques". In: *IEEE Transactions on Industrial Informatics* 9.4 (Nov. 2013), pp. 2256–2263. ISSN: 1551-3203.

[111] H. Akagi. "New trends in active filters for power conditioning". In: *IEEE Transactions on Industry Applications* 32.6 (Nov. 1996), pp. 1312–1322. ISSN: 0093-9994.

[112] J. R. Stack, T. G. Habetler, and R. G. Harley. "Fault classification and fault signature production for rolling element bearings in electric machines". In: *IEEE Transactions on Industry Applications* 40.3 (May 2004), pp. 735–739. ISSN: 0093-9994.

[113] Christian Szegedy et al. "Going Deeper With Convolutions". In: 2015, pp. 1–9. (Visited on 09/17/2018).

[114] Simon Haykin et al. *Neural Networks and Learning Machines Third Edition*. 2009. ISBN: 9780131471399.

[115] Esa Alhoniemi et al. "Process Monitoring and Modeling Using the Self-Organizing Map". In: *Integrated Computer-Aided Engineering* 6.1 (1999), pp. 3–14.

[116] J. Vesanto and E. Alhoniemi. "Clustering of the self-organizing map". In: *IEEE Transactions on Neural Networks* 11.3 (May 2000), pp. 586–600. ISSN: 1045-9227.

[117] Nan Liu, Jinjun Wang, and Yihong Gong. "Deep Self-Organizing Map for visual classification". In: *Proceedings of the International Joint Conference on Neural Networks*. 2015. ISBN: 9781479919604.

[118] Christos Ferles, Yannis Papanikolaou, and Kevin J. Naidoo. "Denoising Autoencoder Self-Organizing Map (DASOM)". In: *Neural Networks* 105 (Sept. 2018), pp. 112–131. ISSN: 0893-6080. (Visited on 12/12/2018).

[119] Thore Graepel, Matthias Burger, and Klaus Obermayer. "Self-organizing maps: Generalizations and new optimization techniques". In: *Neurocomputing* 21.1 (Nov. 1998), pp. 173–190. ISSN: 0925-2312. (Visited on 12/12/2018).

[120] Cenk Budayan, Irem Dikmen, and M. Talat Birgonul. "Comparing the performance of traditional cluster analysis, self-organizing maps and fuzzy C-means method for strategic grouping". In: *Expert Systems with Applications* 36.9 (Nov. 2009), pp. 11772–11781. ISSN: 0957-4174. (Visited on 10/18/2017).

[121] X. Tan et al. "Recognizing Partially Occluded, Expression Variant Faces From Single Training Image per Person With SOM and Soft k-NN Ensemble". In: *IEEE Transactions on Neural Networks* 16.4 (2005), pp. 875–886. ISSN: 1045-9227.

[122] Ritwik Kumar et al. “Maximizing all margins: Pushing face recognition with Kernel Plurality”. In: *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2375–2382. ISBN: 978-1-4577-1102-2.

[123] Ritwik Kumar, Arunava Banerjee, and Baba C. Vemuri. “Volterrafaces: Discriminant analysis using Volterra kernels”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 150–155. ISBN: 978-1-4244-3992-8.

[124] Juha Vesanto. “SOM-based data visualization methods”. In: *Intell. Data Anal.* 3.2 (1999), pp. 111–126. ISSN: 1088-467X.

[125] Laurens Van Der Maaten and Geoffrey Hinton. *Visualizing Data using t-SNE*. Tech. rep. 2008, pp. 2579–2605.

[126] *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. URL: `http://yann.lecun.com/exdb/mnist/` (visited on 11/02/2017).

[127] Alexander Vergara et al. “Chemical gas sensor drift compensation using classifier ensembles”. In: *Sensors and Actuators B: Chemical* 166-167 (May 2012), pp. 320–329. ISSN: 0925-4005. URL: `http://www.sciencedirect.com/science/article/pii/S0925400512002018` (visited on 06/05/2018).

[128] *UCI Machine Learning Repository: Citation Policy*. URL: `https://archive.ics.uci.edu/ml/citation_policy.html` (visited on 06/05/2018).

[129] Peter Lipton. “Contrastive Explanation”. In: *R. Inst. Philos. Suppl.* (), pp. 247–266. ISSN: 1358-2461.

[130] Ernest Adams. “The logic of conditionals. Inquiry (Oslo), vol. 8 (1965), pp. 166197. - Ernest W. Adams. Probability and the logic of conditionals. Aspects of inductive logic, edited by Jaakko Hintikka and Patrick Suppes, Studies

in logic and the foundations of mathemat". In: *J. Symb. Log.* 39.3 (1974), pp. 609–611. ISSN: 0022-4812.

[131] E Erwin, K Obermayer, and K Schulten. "Self-organizing maps: ordering, convergence properties and energy functions." In: *Biol. Cybern.* 67.1 (1992), pp. 47–55. ISSN: 0340-1200.

[132] Aliyu Usman Ahmad and Andrew Starkey. "Application of feature selection methods for automated clustering analysis: a review on synthetic datasets". In: *Neural Comput. Appl.* 29.7 (2018), pp. 317–328. ISSN: 09410643.

VITA

Kasun Amarasinghe received his B.Sc. in Computer Science from the University of Peradeniya in Sri Lanka in 2011. He joined the Doctor of Philosophy program at Virginia Commonwealth University in Richmond, Virginia in 2014. His research experience includes research assistant positions at the University of Idaho, and Virginia Commonwealth University. He has gained research experience by collaborating with universities, U.S. Department of Energy National Laboratories and Industry partners. His research interests are explainable artificial neural networks, pattern recognition, human-machine interaction, and evolutionary algorithms