DEEPEDGE: DEEP-LEARNING BASED IMAGE RECOGNITION SYSTEM ON EDGE COMPUTING INFRASTRUCTURES

BY

CHANG LIU
B.E. HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (2011)
M.S. UNIVERSITY OF MASSACHUSETTS LOWELL (2018)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL

AUGUST, 2019

Dissertation Supervisor
Yu Cao
Ph.D. Professor
Department of Computer Science
University of Massachusetts Lowell

Thesis Committee Members
Benyuan Liu
Ph.D. Professor
Department of Computer Science
University of Massachusetts Lowell

Yan Luo
Ph.D. Professor
Department of Electrical and Computer Engineering
University of Massachusetts Lowell

DEEPEDGE: DEEP-LEARNING BASED IMAGE RECOGNITION SYSTEM ON EDGE COMPUTING INFRASTRUCTURES

BY

CHANG LIU
B.E. HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (2011)
M.S. UNIVERSITY OF MASSACHUSETTS LOWELL (2018)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL

Chang Liu

Chang Liu

07/29/2019

Date

Dissertation Supervisor

[illegible]

Yu Cao, Ph.D. Professor
Department of Computer Science
University of Massachusetts Lowell

Thesis Committee Members

[illegible]

Benyuan Liu, Ph.D. Professor
Department of Computer Science
University of Massachusetts Lowell

[illegible]

Yan Luo, Ph.D. Professor
Department of Electrical and Computer Engineering
University of Massachusetts Lowell

DEEPEDGE: DEEP-LEARNING BASED IMAGE RECOGNITION SYSTEM ON EDGE COMPUTING INFRASTRUCTURES

BY

CHANG LIU

ABSTRACT OF A DISSERTATION SUBMITTED TO THE FACULTY OF THE
DEPARTMENT OF COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE
UNIVERSITY OF MASSACHUSETTS LOWELL

AUGUST, 2019

ABSTRACT

Deep learning, aimed to learn multiple levels of representation and abstraction that help infer knowledge from data such as images, video, audio and text, is making astonishing improvements in computer vision, speech recognition, multimedia analysis, and medical imaging. While promising, current progress in deep learning algorithm and system fail to meet the hardware requirement and time constraint. Most of the complex deep learning models can only be deployed in powerful super-computers and cloud servers equipped with GPU and CPU, making it inaccessible to various mobile devices. However, in real world, the need for deploying high-accurate, fast-response deep learning models in mobile devices is increasing. How to build a high-efficient system with limited computation resource while maintaining the high accuracy and low latency is a becoming a more and more challenging task in system research and engineering field.

This thesis proposes a deep learning based system (DeepEdge) running on edge computing infrastructures to solve such problems. We begin our research by training convolutional neural network (CNN) for fine-grained categories on various dataset including general images (food, vegetable) and medical images (chest X-ray images). Then we explore the deployment on edge devices, especially for mobile devices running Android. We implement various CNN models with detailed evaluation for their response time and memory consumption. Additionally, we study two deployment mechanisms using C/S paradigm and stand-alone client paradigm. Our system have shown outstanding performance in these two paradigms, especially in the following two aspects: (1) the deep learning-based visual image recognition algorithm achieves the best-in-class recognition accuracy; (2) the proposed image recognition system employing edge computing paradigm overcomes some inherent problems of traditional mobile cloud computing paradigm, such as unacceptable system latency and low battery life of mobile devices. We have conducted extensive experiments with real-world data. Our results have shown that the proposed system achieves three objectives: (1) outperforming existing work in terms of image recognition accuracy; (2) reducing response time that is equivalent to the minimum of the existing approaches; and (3) lowering energy consumption which is close to the minimum of the state-of-the-art.

We also investigate the application in different scenarios, especially in the healthcare domain. We first study the application in dietary assessment and developed novel algorithm and application that can help monitor people's food intake and calorie information. Second, we expand our research focus and apply our system in mobile-based tuberculosis (TB) diagnosis and deploy our mobile application and system in Perú for field study in real world settings. At last, we collaborate with retail industry and develop system in self-service supermarkets for automatic vegetable and fruit recognition. Our system has proven stable performance in such applications and shown broad potential in healthcare and retail industries.

ACKNOWLEDGEMENTS

Pursuing a Ph.D degree is never an easy journey. It's like a marathon that requires a lot of attention, patience and persistence. I could never have reached the heights without the help, support and guidance of a lot of people.

First of all, I would like to thank my supervisor, Dr. Yu Cao, for his devotion to my research throughout the past five years. Dr. Cao has long been a strong supporter and coach in my Ph.D study. He has supported me not only by providing a research assistantship over the past five years, but also academically and emotionally through the rough road to finish this thesis. He helped me come up the thesis topic, guided me over the experiments and gave me directions to solve the difficulties along the journey. Without his help, this dissertation could never be finished.

I'm also very grateful to my Ph.D. committee members, Dr. Benyuan Liu and Dr. Yan Luo for their time, academic support and valuable guidance in general. Dr. Liu has been a talented researcher and is always providing insightful advice in my research. I'd also like to thank him for his patient help in revising my papers and guiding me through the projects. I'm also grateful for the support of Dr. Yan Luo. He has been a very good collaborator and patient helper that always supports me, especially on his mobile computing projects.

Over the past five years, I am fortunate to have the opportunity to work with a group of energetic and talented schoolmates (Ning Zhang, Peng Hou, Qilei Chen, Ying Li, Dechun Wang, Xizhe Wang, Chenxi Zhang, Pengfei Zhang, Xinzi Sun, Jing Ni, Lijian Wan, Wenjing Yang, Baochen Sun etc) in the department of computer science in University of Massachusetts Lowell. I enjoyed every moment that we have worked together, especially the debugging night in the labs. I would also thank Marlon Alcantara and Terry Griffin for their support in my medical research projects. I appreciate the friendship and all their encouragement to finish this dissertation.

I would also show my deep thanks to the computer science secretary Karen Volis, system manager Tuyen Nguyen and other professors for their assistance throughout the five years. Your assistance makes the life easier.

I also want to express my heartfelt thanks to friends in the Chinese Christian student fellowship, Raymond & Gail, Alsa & Rick, Augustine & Sharon, Jinyi Chen, Junwei Huang, Wei Ye, Jonny Wong, Peilong Li & Jingwen Wang, Yujia Zhou, Shan Cao, Qing Yu, Zheng Li, Bin Tan, Ting Wang and Elizabeth Jin. I'm also thankful for Dr. Chunxiao (Tricia) Chigan, Ibukun Dada, Jimmy Oladimeji from Fusion Church for helping me in seeking God. I also want to thank the Pastors Benjamine Tu and Gideon, the brothers and sisters from Chinese Bible Church of Great Lowell (CBCGL) for their endless love and support. Thank God for his provision and faithfulness.

Last but not least, I must thank my mother Xingzhen Wang, my brother Bin Liu for their unconditional support and love. I'd like to thank my father for his deep love and support during his life. I know he must be very happy for me in the heaven.

Dedicated to my parents

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

INTRODUCTION

1.1 Problem Statement

In the last few years, we have witnessed an explosive increase of mobile and wearable computing devices (e.g., the smart watch and smart phone) in the consuming electronics market. One common characteristic of these devices is that many of them have inexpensive, unobtrusive and multi-modal sensors. These sensors enable us to collect multimedia data (e.g., video, audio and image) in natural living environments. Due to the ubiquitous nature of mobile and wearable devices, it is now possible to use these devices to develop pervasive, automated applications for computer vision tasks like image classification and object detection. One example of such application is to use mobile devices as a pervasive journal collection tool and to employ cloud service as a data analysis platform. The combination of mobile device and cloud service could contribute to improving the accuracy of image recognition.

While promising, one of the major barriers of adopting automatic recognition system into practice is how to design and develop effective and efficient algorithms and system to derive the information (e.g., image type, class label and region information) from input images. As shown in Figure 1.1, there are three factors (e.g, accuracy, battery and latency) that limit the development of image recognition system. Considering the limited computation resources and low battery life on mobile device, it is more challenging to develop such a system within the mobile cloud computing paradigm. We have carefully investigated this problem and have identified two major challenges. The first major challenge is how to design effective and efficient analytic algorithms to achieve optimal recognition accuracy. The second major challenge is how to develop a system that can minimize energy consumption and response time.

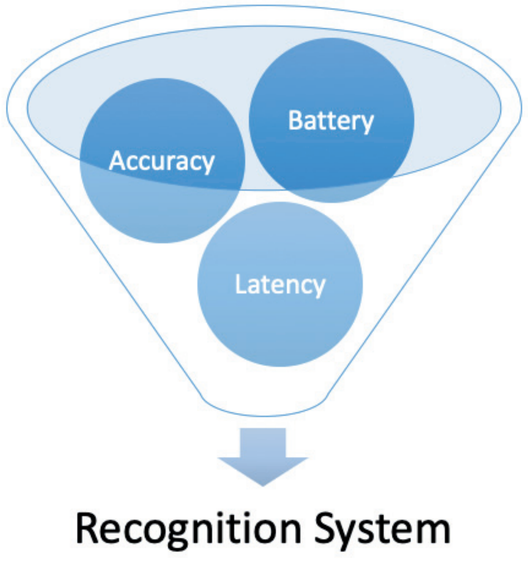


Figure 1.1: Challenges for developing image recognition system on mobile devices.

1.2 Proposed Approach

To address the first issue (recognition accuracy), we plan to develop new deep learning-based algorithms. Deep learning (also known as representation learning, feature learning, deep structured learning, or hierarchical learning) is a new area of machine learning research. It allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. In the last five years, these techniques have improved the state-of-the-art in speech recognition, computer vision, natural language processing, and many other domains. Our extensive experiments in this thesis have shown that, compared with traditional hand engineered features (e.g., SIFT) and shallow learning-based classification algorithms (e.g., Support Vector Machine (SVM)), our proposed deep learning-based classification algorithms could improve the recognition accuracy substantially. We also developed other image analysis algorithms to enhance the image quality for data analysis. All these algorithms have been integrated into an edge computing based real-time computing system.

To address the second issue (energy consumption and response time), we aim to design and employ a real-time recognition system employing edge computing service paradigm. The proposed system distributes the data analytics throughout the network by splitting the recognition task between the edge devices (close to end users) and the servers (in the cloud). Edge computing refers to the enabling technologies that allow computation to be performed at the edge of the network in a stream fashion. Edge computing is a non-trivial extension of cloud computing from the core network to the edge network. The proposed edge computing service infrastructure is particularly useful for our application because most of the mobile devices have limited computation capacity and battery life. Hence, it is difficult for them to support computational-intensive tasks. At the same time, our proposed image analysis algorithms usually involve heavy computation and may require much more computation resources.

In this thesis, we focus on two major research efforts. The first research effort aims to develop new recognition algorithms, including new image recognition algorithms based on deep learning and image pre-processing and segmentation algorithms to enhance the quality of image. The second research effort aims to design a real-time recognition system. The proposed system employs edge computing service paradigm and distributes the data analytics throughout the network. Specifically, the proposed system will split the recognition tasks between the edge devices (which is physically close to the user) and the server (which is usually located in the remote cloud). For example, in our system, the edge devices (e.g., users smart phone) can perform light-weight computation on image for recognition. Then, our system will transmit the images (after the light-weight computation at edge device) to the server in the cloud to perform more accurate recognition tasks. By distributing the analytics throughout the network, our system can achieve significant improvement in the recognition accuracy, while minimizing the response time and energy consumption. In this thesis, we implemented a prototype system to verify our hypothesis and evaluate the proposed algorithms. Our prototype runs on both edge device and server. We also conducted extensive experiments with real-world data. The results show that our system achieves very impressive results on the following three aspects. First, to the best of our knowledge, the recognition accuracy using our proposed approach outperformed all other reported results. Second, the response time of the proposed system is equivalent to the minimum of the existing approaches. Last but not the least, the energy consumption of the proposed system is close to the minimum of the state-of-the-art.

1.3 Organization

The remainder of this thesis is organized as follows. Chapter 2 provides background on image recognition, deep learning-based image recognition and mobile-based computing system. Chapter 3 introduces the current state-of-art of relevant research. In the next three chapters, we discuss our proposed approach in three different application scenarios, including image recognition in general images and medical images with algorithm design and implementation in edge-computing devices. Chapter 4 presents the food recognition and dietary assessment research. Methods for medical image analysis and tuberculosis diagnosis are discussed in Chapter 5. We introduce the algorithm and system in general image recognition for vegetable and fruit image recognition in real-world retail stores in Chapter 6. Finally, we offer our concluding remarks and future work in Chapter 7.

Chapter 2

BACKGROUND ON IMAGE RECOGNITION SYSTEM

Image recognition refers to the technologies that can recognize certain people, animals, objects or other targeted subjects through the use of algorithms and machine learning techniques. It's becoming a very popular topic in computer vision and has broad application in surveillance, autonomous driving, healthcare, social media and recommendation. There are several kinds of sub-tasks in such field, for example, image classification, object detection and localization. As shown in Figure 2.1, these three types of tasks have their own different approaches and targets. Image classification usually refers to processing the whole image level information, the final goal is to give a label to the whole image; Image localization has a different target, requiring to find the objects in the precise location and predict its corresponding label; For object detection, the bounding box is also predicted to get the precise boundary of the objects in the picture, with the possibility that each image may have multiple objects.

Figure 2.2 shows the pipeline for a traditional image recognition task. An input image is first fed into an image recognizer. The image recognizer will process

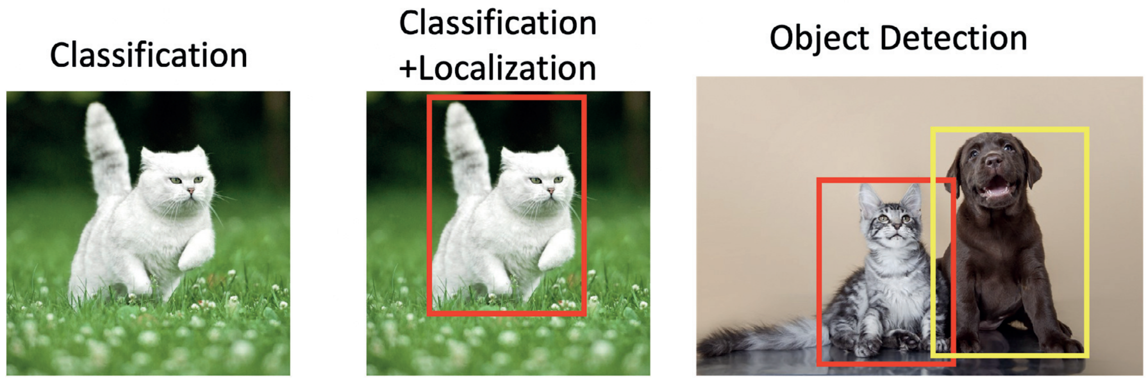


Figure 2.1: General image recognition tasks.

this image and generate the output for the image. For an image classification task, the recognizer is an image classifier that classify the image and predict the label. For an object detection task, the recognizer is an object detection that detects the precise bounding box of the objects and predicts the corresponding the coordinates. In most cases, the image recognizer is trained by using some image features from the training image set.

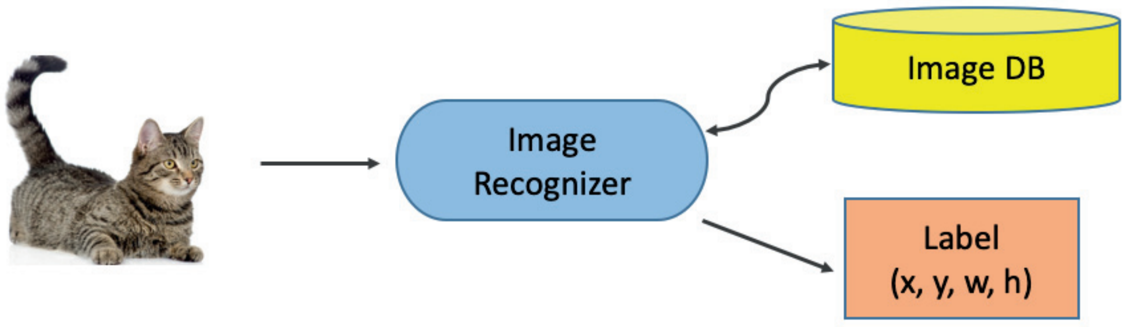


Figure 2.2: Processing pipeline for an image recognition task.

As shown in Figure 2.3, there are typically two categories of features: 1) hand-engineered features that encode the global or local information from the image. For example, Shape Matrices, Invariant Moments [2], Histogram Oriented Gradients (HOG [3]) and Co-HOG are some examples of global descriptors. SIFT [4], SURF [5], LBP [6], BRISK [7], MSER [8] and FREAK [9] are some examples of local descriptors.

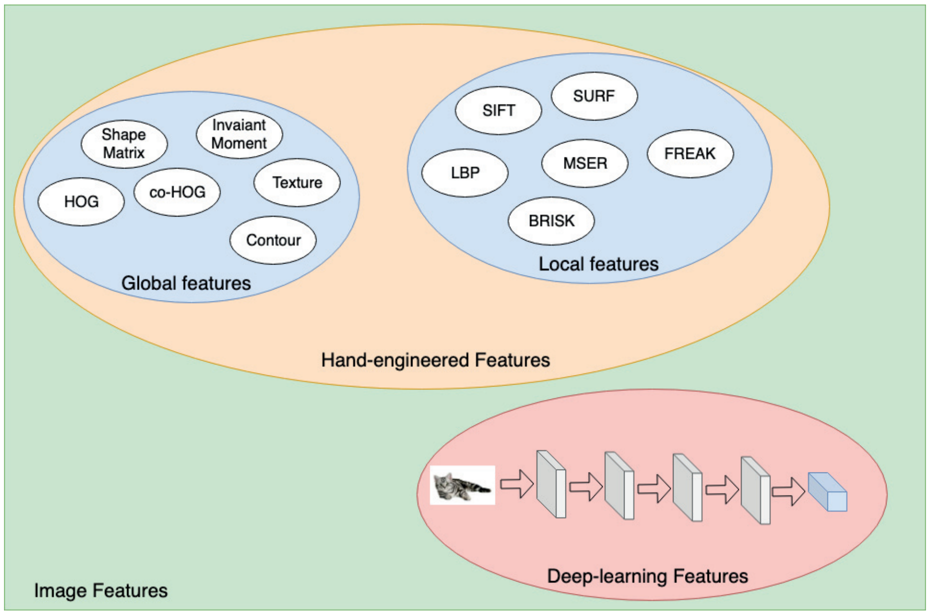


Figure 2.3: Two categories of image features.

These feature descriptors encode the image using their fixed patterns regarding color, texture, shape and others to represent the inherent unique spatial information. 2) deep learning-based image features that are extracted from image automatically using neural network model. For this kind of image features, since the model is learned from a large-scale dataset, there are no fixed rules designed by human to represent image. Those features have shown outstanding classification and detection performance in major recognition leader board [10, 11, 12]. Most neural network models are consist of different layers, including convolutional layer, pooling layer, ReLU layer and fully-connected layers with different activation function and connection structure. Such network design introduces significant improvement over simple model structure and enhances the capability of feature representation over hand-engineered features.

In this following section, more discussions about the specific challenges and research problems will be covered to give more descriptions about the background on this image recognition problem.

2.1 Feature-based Image Recognition

In image recognition community, the popular methods have long been dominate by the feature-based algorithms. The method of finding image displacements which is easiest to understand is the feature-based approach. This finds features (for example, image edges, corners, and other structures well localized in two dimensions) and compares the feature maps using mathematical distances and similarities. This feature-based method (FBR) usually follows such routines: 1) first, it proceeds by computing a number of properties of the input image and combining them into a feature vector; 2) second, an object model is constructed by a set of feature vectors associated with a set of representative images for that object; 3) at last, a new image is classified by computing its feature vector and compares it with the model vector. An image is classified as an instance of the object when the object model contains the feature vector that is closest to the image feature vector.

Research activities in this category focus on developing different types of visual features and classification algorithms to score different types of object categories. Most of the papers employ texture features (e.g., Local binary patterns (LBP) [13, 14], Daubechies wavelets [15]) or geometry features (e.g., circularity, Hessian shape features [16]). The classification algorithms employed in these papers range from simple threshold-based approach or k-nearest neighbors (K-NN) algorithm to more complicated methods, such as Decision tree and Support Vector Machine (SVM); The second category of related work is focusing on image categorization on the region level [17]. The main stream methodology in this area is based on local patch representation of the image content (e.g., visual bag of words (Visual BoW) approach). This type of dense sampling of simple features are then fed to non-linear kernel-based classifier, such as SVM classifier. The ultimate goal of this research effort is to discriminate between object and its background.

2.2 Deep learning-based Approach for Recognition

Deep learning, as shown in Figure 2.4 aims to learn multiple levels of representation and abstraction that help infer knowledge from data such as images, videos, audio, and text, is making astonishing gains in computer vision, speech recognition, multimedia analysis, and drug designing [18]. Briefly speaking, there are two main classes of deep learning techniques: purely supervised learning algorithms (e.g., Deep Convolutional Network), unsupervised and semi-supervised learning algorithms (e.g., Denoising Autoencoders [19], Restricted Boltzmann Machines, and Deep Boltzmann Machines [20]). With the help of large-scale and well-annotated dataset like ImageNet, it's now feasible to perform large scale supervised learning using Convolutional Neural Network (CNN). The issue of convergence has been addressed by Hinton's work in 2006. Subsequent theoretical proof and experimental results both shows that large scale pre-trained models in large domain, with specific small scale unlabeled data in another domain, will give excellent result in image recognition and object detection. To address the issue of limited abilities of feature representation, many researchers have proposed more complex CNN network structure, like VGG [21], ZFNet [22], GoogLeNet [12] and so on. On the other hand, ReLU [23] is also proposed to make it converge faster and also gains a better accuracy. Most of current researchers have put efforts in making the network deeper and avoid saturation problem.

2.3 Mobile-based Computing System

There has been a significant amount of research going on and plenty of them focus on studying the system performance of deep learning system running on the mobile devices. Previous work on system research focus on the overall evaluation metrics when conducting the recognition tasks, which include accuracy, inference time, response time and power consumption [24]. The studied system can be divided into

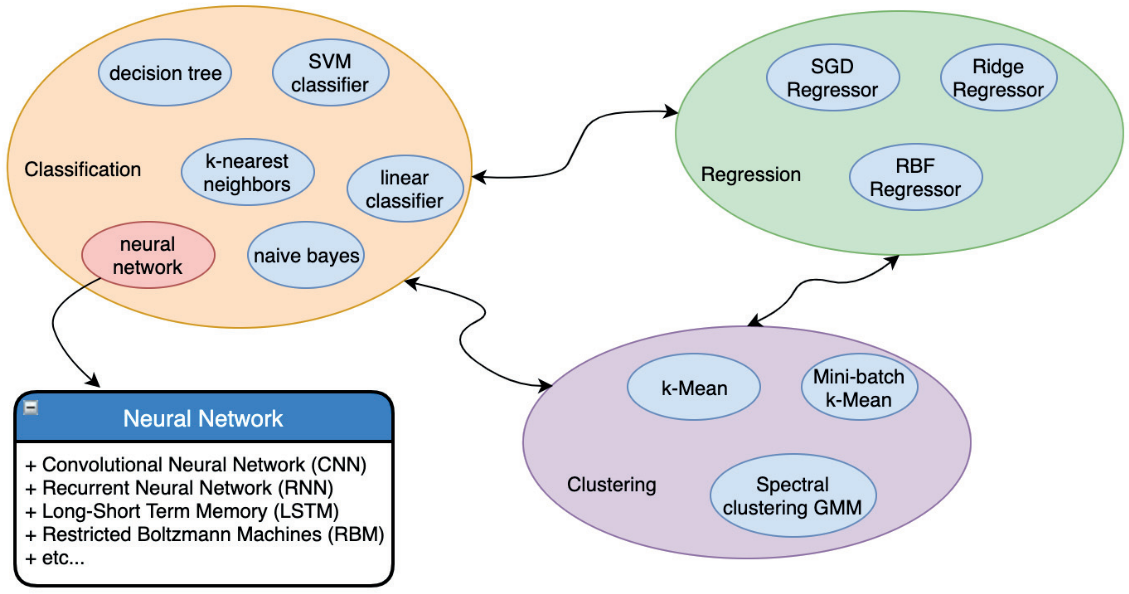


Figure 2.4: Machine learning algorithms.

two categories. The first category focus on studying the combination of cloud-based server and mobile devices [25, 26]. Researchers developed the system using mobile and cloud server together. Chen et al. [25] proposed to use mobile-edge and cloud services for system integration and studied the influence of computation offloading for multiple-users. Kang et al. [27] designed a lightweight scheduler to automatically partition DNN computation between mobile devices and data centers at the granularity of neural network layers to reduce latency and power consumption. The second category focus on developing system without cloud intervention. Keiji and Austin et al. developed a system [28, 29] for food recognition using CNN architecture and running the inference on-device separately. Latifi et al. [30] designed a system called CNNDroid for running CNN models on Android devices with GPU-accelerated execution. The device-based mobile approaches studied the system performance in real-world scenarios and provide complete evaluation and guideline for deployment.

Edge computing [31] usually refers to the enabling technology that allows computation to be performed at the edge of the network. The "edge" devices can be any computing and network resources along the path between data sources and cloud data center. The data source can be any sensing devices like smartphone, smartwatch, PDA and tablet that collect sensor data like image, audio and video. Cloud data center is equipped with powerful servers that can perform complex computation and data processing. By utilizing the computation ability of edge devices, we can address the critical issues of response time requirement, battery life constraint, bandwidth cost saving, as well as data safety and privacy. The major challenge [26] in applying deep learning algorithms and building visual categorization system is to devise a high-performance mechanism that utilizes the edge computing power for accurate recognition within limited response time and computation resources.

Chapter 3

CURRENT STATE-OF-ART OF RELATED RESEARCH

Despite a large body of knowledge in image recognition, very little research has been conducted to study the performance of deep learning-based system in mobile devices, or to balance between the system accuracy and the latency, power consumption and hardware constraints. Another related research efforts are in the area of medical image analysis, especially in the field of tuberculosis diagnosis. Even though much efforts are put in general feature engineering, few of them use deep learning methods, especially for tuberculosis detection. In this section, we will discuss the current state-of-art methods of related work in image recognition, medical image analysis and edge computing.

3.1 Image Recognition and Object Detection

Object detection and image recognition is one of the fundamental problems in computer vision. Much of previous work has focused on extracting features from the images followed by matching or classification algorithms. These feature-based methods can be divided into two categories. The first category is to use template matching that search over the image of interest (IoU) to identify the object and measure the similarity of the template with the regions. Common features are used to measure such similarities, including cross-correlation coefficient [32], Fourier descriptor [33] and texture features [34]. The second category is called model-based approaches, focusing on searching the correspondences between the model and image features. Unlike the simple template matching that searches through all the possible candidates, model-based approach use various heuristics to guide and improve the search. Typical example include the tree-based search model [35], alignment-based method [36].

However, in recent years, with the development of large scale, well-annotated dataset and increasing computation capability equipped with CPU and GPU, deep learning based approaches show excellent performance comparing with the previous feature-based methods. More and more neural network structures are devised to speed up the training process and improve the accuracy [1, 10, 12, 21]. Such optimization focus on introducing more neural network layers [12], reducing the abundant computation neurons [37], adding noise in activation function [23] and introducing non-linearity to avoid saturation [38]. In the recent public leader board of ImageNet challenge, deep learning based CNN model and approaches have surpassed all existing feature-based approaches by a large margin. The rational behind such difference is that deep learning-based approach can learn the inherent features that is not hard coded as human engineered features, these improvement can encode more image information in the feature map and boosts the representation ability.

3.2 Edge-based Image Recognition System

Recent years have witnessed the vast progress in edge devices [31, 39, 40]. Under the edge computing infrastructure, part of the data processing tasks may be pushed to the edge of the network. One of the core idea is called "Collaborative Edge" [31], which refers to the architecture that connects the edges of multiple stakeholders. These stakeholders may be geographically distributed and they may have distinct physical location and network structure. Under this infrastructure, the cloud paradigm is extended to the edge of the network. Therefore, such an edge computing service infrastructure offers a unifying paradigm for cloud-based computing and Internet of Things (IoT)- based computing. It has the potential to address the issues of delayed response time, reduced battery life, limited bandwidth, and data security and privacy. However, most of the existing use cases of edge computing-based image recognition applications [41, 42] are relatively simple examples with small data sets. Novel user cases and intriguing applications with more challenging tasks, such as larger data sets and sophisticated computation, are needed for evaluating the efficacy and effectiveness of edge computing in various tasks, especially for the time consuming applications that embeds with complex deep learning model and requires long processing time. In such application [26], more computation offloading is proposed and evaluated to achieve better accuracy within limited time.

Edge computing could yield many benefits. Most of the edge computing systems have shown fast response time and reduced energy consumption. Comparing with traditional cloud computing paradigm, edge computing can utilize the edge device's computation capability and reduce the network transmission time. For example, researchers [43] show that using cloudlets to offload computing tasks can improve the response time between 80 and 200 ms and reduces energy consumption by 30 to 40 percent. Security and privacy are another two benefits. By conducting computation on the edge devices, the data is preprocessed and sensitive information is eliminated to protect the user's privacy. In most of existing healthcare and edge computing application [41, 44], the data is preprocessed and transferred to the cloud devices by offloading some computation in the edge side, preserving the important feature information and reducing the abundant and sensitive information simultaneously. The development of edge computing system in healthcare domain using the existing cloud-based service infrastructures, has began to shown its significance in reducing system latency, improving accuracy and minimizing resource consumption.

3.3 Deep Learning in Medical Imaging

In medical image research community, especially tuberculosis (TB) diagnosis in X-ray images, due to the limited size of dataset, it's very hard to train a CNN model for accurate classification. The research focus in such field can be divided into two categories. The first one is to build a large scale, well-annotated dataset. While there are some evaluation efforts in TB screening tests on developing countries, to the best of our knowledge, there is no large-scale, real-world, well-annotated, and public available X-ray image database dedicated for TB screening diagnosis. Most of the existing research in the area of computer-aided TB screening employed small data sets for evaluation and validation. Most of the datasets have less than 200 images. There are a few large data sets, such as ImageCLEF, JSRT Digital Image Database, and ANODE Grand Challenge Database, have over tens of thousands images. However, they only include one or two aspects of TB manifestations (e.g., pulmonary nodule). Without a large scale data sets with high qualify annotation, it will be very difficult to determine the efficacy of existing and proposed approach when applied to real-world clinic data. Furthermore, dedicated image annotation software tools and database storage software that can support the manipulations of the X-ray images are needed to facilitate the image annotation and image management.

Second, much efforts are put into designed good deep learning models for accurate classification. Even though CNN and other deep learning can have good performance in general image classification tasks, it's still very hard to apply the

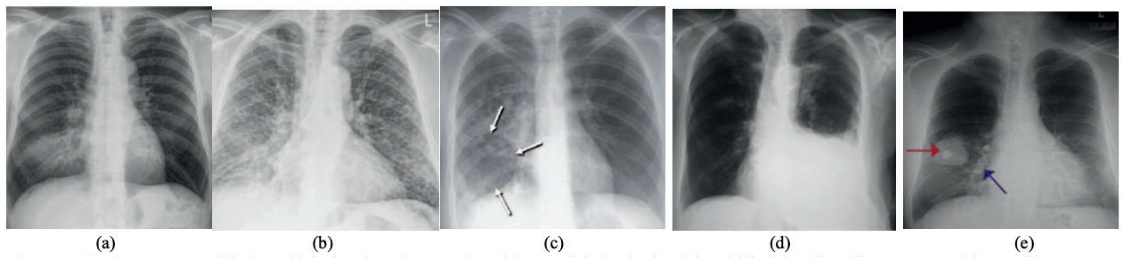


Figure 3.1: Various TB manifestations with minor differences in the same category.

deep learning model into the medical dataset directly, due to the minor differences and various kinds of confusion and noise in the images. As shown in Figure 3.1, these TB manifestations are very hard to distinguish and find the differences, even for the human annotator and doctors. In modern deep learning based medical imaging models [45, 46], several network structure and training strategy are proposed to eliminate the dependence on the dataset size and image quality. For example, Ronneberger et al [47] proposed a network called UNet to combine CNN layers with up-sampling layers for image segmentation. These networks modified the kernel size and sampling methods, combine multiple layers and encode more layers' feature maps, such optimize has shown excellent performance in boosting the accuracy.

Chapter 4

FOOD RECOGNITION AND DIETARY ASSESSMENT

Literature has indicated that accurate dietary assessment is very important for assessing the effectiveness of weight loss interventions. However, most of the existing dietary assessment methods rely on memory. With the help of pervasive mobile devices and rich cloud services, it is now possible to develop new computer-aided food recognition system for accurate dietary assessment. However, enabling this future Internet of Things-based dietary assessment imposes several fundamental challenges on algorithm development and system design. In this chapter, we set to address these issues from the following two aspects: (1) to develop novel deep learning-based visual food recognition algorithms to achieve the best-in-class recognition accuracy; (2) to design a food recognition system employing edge computing-based service computing paradigm to overcome some inherent problems of traditional mobile cloud computing paradigm, such as unacceptable system latency and low battery life of mobile devices. We have conducted extensive experiments with real-world data. Our results have shown that the proposed system achieved three objectives: (1) outperforming existing work in terms of food recognition accuracy; (2) reducing response time that is equivalent to the minimum of the existing approaches; and (3) lowering energy consumption which is close to the minimum of the state-of-the-art.

4.1 Introduction

In the US, more than one-third (34.9% or 78.6 millions) of adults are obese and approximately 17% (or 12.7 millions) of children and adolescents aged 2 to 19 years are obese [48]. There were more than 1.9 billion adults, 18 years and older, were overweight on earth in 2014 [49]. Documenting dietary intake accurately is crucial to help fight obesity and weight management. Unfortunately, most of the current methods for dietary assessment (for example, 24 hour dietary recall [50] and food frequency questionnaires [51]) must rely on memory to recall foods eaten.

In the last few years, we have witnessed an explosive increase of mobile and wearable computing devices (e.g., the smart watch and smart phone) in the consuming electronics market. One common characteristic of these devices is that many of them have inexpensive, unobtrusive and multimodal sensors. These sensors enable us to collect multimedia data (e.g., video and audio) in natural living environments. Due to the ubiquitous nature of mobile and wearable devices, it is now possible to use these devices to develop pervasive, automated solutions for dietary assessment [52, 53, 54, 55]. One example of such solutions is to use mobile devices as a pervasive food journal collection tool and to employ cloud service as a data analysis platform. The combination of mobile device and cloud service could contribute to improving the accuracy of dietary assessment. As a result, in the last few years, we have seen several mobile cloud software solutions to improve the accuracy of dietary intake estimation. One common issue among these solutions is that the users of the software must enter what they have eaten manually. To address this issue, visual-based food recognition algorithms and systems have been proposed [53, 54]. A recent review by Martin et al. [56] also indicated that using digital imaging techniques for food recognition is superior to many other methods of dietary assessment techniques. Some advantages of visual-based food recognition systems include: reduced burden for users to recall the food, improved accuracy and efficiency of dietary recall.

While promising, one of the major barriers of adopting automatic dietary assessment system into practice is how to design and develop effective and efficient algorithms and system to derive the food information (e.g., food type) from food images. Considering the limited computation resources and low battery life on mobile device, it is more challenging to develop such a system within the mobile cloud computing paradigm. We have carefully investigated this problem and have identified two major challenges. The first major challenge is how to design effective and efficient analytics algorithms to achieve optimal recognition accuracy. The second major challenge is how to develop a system that can minimize energy consumption and response time.

To address the first issue (recognition accuracy), we plan to develop new deep learning-based algorithms. Deep learning [57, 58] (also known as representation learning, feature learning, deep structured learning, or hierarchical learning) is a new area of machine learning research. It allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. In the last five years, these techniques have improved the state-of-the-art in speech recognition, computer vision, natural language processing, and many other domains. Our extensive experiments in this chapter have shown that, compared with traditional hand engineered features (e.g., SIFT [4]) and shallow learning-based classification algorithms (e.g., Support Vector Machine (SVM)), our proposed deep learning-based classification algorithms could improve the recognition accuracy substantially. We also developed other image analysis algorithms to enhance the food image quality for data analysis. All these algorithms have been integrated into an edge computing based real-time computing system, which is discussed in the next paragraph.

To address the second issue (energy consumption and response time), we aim to design and employ a real-time food recognition system employing edge computing service paradigm. The proposed system distributes the data analytics throughout the network by splitting the food recognition task between the edge devices (close to end users) and the servers (in the cloud). Edge computing refers to the enabling technologies that allow computation to be performed at the edge of the network in a stream fashion. Edge computing is a non-trivial extension of cloud computing from the core network to the edge network [43, 59, 31, 39, 40, 60, 61]. The proposed edge computing service infrastructure is particularly useful for our application because most of the mobile devices have limited computation capacity and battery life. Hence, it is difficult for them to support computational-intensive tasks. At the same time, our proposed food image analysis algorithms usually involve heavy computation and may require much more computation resources.

In this chapter, we focus on two major research efforts. The first research effort aims to develop new food recognition algorithms, including new food image recognition algorithms based on deep learning and image pre-processing and segmentation algorithms to enhance the quality of food image. The second research effort aims to design a real-time food recognition system for dietary assessment. The proposed system employs edge computing service paradigm and distributes the data analytics throughout the network. Specifically, the proposed system will split the food recognition tasks between the edge devices (which is physically close to the user) and the server (which is usually located in the remote cloud). For example, in our system, the edge devices (e.g., users smart phone) can perform light-weight computation on food image for food recognition. Then, our system will transmit the food images (after the light-weight computation at edge device) to the server in the cloud to perform more accurate recognition tasks. By distributing the analytics throughout the network, our system can achieve significant improvement in the recognition accuracy, while minimizing the response time and energy consumption. In this project, we implemented a prototype system to verify our hypothesis and evaluate the proposed algorithms. Our prototype runs on both edge device (Xiaomi Note, running Android 6.0.1 marshmallow) and server (an in-house GPU cluster). We also conducted extensive experiments with real world data. The results show that our system achieves very impressive results on the following three aspects. First, to the best of our knowledge, the food recognition accuracy using our proposed approach outperformed all other reported results. Second, the response time of the proposed system is equivalent to the minimum of the existing approaches. Last but not the least, the energy consumption of the proposed system is close to the minimum of the state-of-the-art.

The rest of the chapter is organized as follows. In Section 4.2, we introduce related work in computer-aided dietary assessment, visual-based food recognition, deep learning, and edge computing. In Section 4.3, we present the architecture, components, and algorithms for the proposed system based on deep learning and edge computing. In Section 4.4, we describe the implementation details of our system. Section 4.5 presents the evaluation results, which include recognition accuracy, power consumption, response time, etc. Section 4.6 discusses the system limitations. In Section 4.7, we make concluding remarks.

4.2 Related Work

Estimating dietary intake accurately with a high-quality food journal is crucial for managing weight loss [62]. Unfortunately, due to many technical barriers, how to improve the accuracy of dietary intake estimation is still an open question. In this chapter, we aim to develop a systematic approach as a first step to address this issue.

We envision that there are four most relevant research areas, listed as below.

The first related research area is to enhance the accuracy of diet assessment with computer-aided solutions. Due to the recent advances in electronics, it is now possible to develop computer-aided solutions to transform healthcare from reactive and hospital-centered to preventive, proactive, evidence-based, person-centered. Dietary assessment is one such area that has gained a lot of attentions from both academia and industry. Among thousands of existing mobile cloud health software and hardware, we have seen many of them (e.g., MyFitnessPal, MyNetDiary, and FatSecret) are dedicated for improving the accuracy of dietary estimates. However, all these applications require the user to enter everything they ate manually. To address this issue, several applications have been developed to improve the level of automation. For example, a recent App entitled "Meal Snap" aims to reduce human efforts by asking the user to take a picture, enter some quick information such as whether user is eating breakfast or lunch, and add a quick text annotation if the user wants to. Unfortunately, the accuracy of calorie estimation is heavily dependent on the accuracy of the manually entered text from user. Therefore, the accuracy is very unstable. Another example of such application is named "Eatly". This application requires the user to take the food image and then rates the food into one of the three categories ("very healthy", "it's OK.", and "unhealthy"). However, the actual rating is performed manually by the community, which consists of the users of this App. In this chapter, we propose new algorithms and system that can recognize the food images (captured by the user with their mobile devices) automatically. This automation reduces the users burden substantially.

The second related research area is to perform dietary analysis using food images and/or videos. In one paper [53], researchers proposed an approach to combine a learning method (manifold ranking-based techniques) and a statistics method (co-occurrence statistics between food items) to recognize multiple food items. In another study [54], the authors proposed a method for fast food detection by researching the relative spatial relationships of local features of the ingredients and a feature fusion technique. NIH also funded a project named "Technology Assisted Dietary Assessment (TADA)". Researchers under this project have investigated different aspects of computer-aided dietary assessment, such as food item recognition, mobile interface design, and data development for food images. They have published several papers on food image recognition [55, 63, 61]. Most of the existing visual-based food recognition algorithms employed traditional signal processing with hand-engineered features (e.g., SIFT [4], HOG [3]) and shallow machine learning algorithms (e.g., SVM). Only very recently, with the striking success of deep learning, people started to research the application of deep learning for food image recognition [29]. Deep learning has the potential to address one main issue associated with existing techniques, which is that the hand engineered features may be useful for screening a few categories of food item but are unable to generalize to other food types. The proposed approach in this chapter is also based on recent advances in deep learning. Related work in deep learning is introduced in the next paragraph.

The third related field is deep learning, which is a branch of machine learning. It allows the computers to learn from experience and understand the world in terms of a hierarchy of concepts using a deep graph with multiple processing layers. Each concept is defined in terms of its relation to simpler concepts [57]. Essentially, deep learning is trying to solve the central problem in representation learning by introducing representations that are expressed in terms of other, simpler representations [57]. It has already been proven useful in many disciplines, such as computer vision, speech recognition, natural language processing, bioinformatics, etc. There are two main classes of deep learning techniques. The first class is purely supervised learning algorithms, such as Deep Convolutional Neural Network (CNN). The second class is unsupervised and semi-supervised learning algorithms, such as Denoising

Auto-encoders and Deep Boltzmann Machines. In this chapter, we focus on deep Convolutional Neural Network (CNN) [10]. Our proposed approach is rooted from CNN and it belongs to the category of supervised learning algorithms. CNNs are biologically inspired [64] (animal visual cortex) variants of Multilayer Perceptrons (MLPs). It is consisted of neurons that have learnable weights and biases. Compared with MLPs, CNN has several distinct features. First, by enforcing a local connectivity pattern between neurons of adjacent layers, CNN could exploit spatially local correlation. Second, each filter in CNN is replicated across the entire visual field, which share the same parameters (e.g., weight vector and bias). Third, the neurons are arranged in three dimensions (width, height, and depth). Furthermore, a feature map can be generated by repeated application of a function across sub-regions of the whole image. Early implementation of CNNs, such as LeNet-5 [1], has been successfully applied to hand writing digital recognition. However, due to the lack of large scale labeled data and limited computation power, CNNs failed to address more complex problems. With the help of large-scale and well-annotated dataset like ImageNet [65], new computing hardware such as graphics processing unit (GPU), and several algorithms advancements such as Dropout [37], it is now possible to train large scale CNNs for complex problems. Recently, many research, such as VGGNet [21], ZFNet [22], GoogLeNet [12], Residual Network [66], has been proposed to address the issue of limited abilities of feature representation. One common strategy is to make the network deeper and avoid saturation issues. Our proposed approach was directly inspired by CNN work from LeNet-5 [1], AlexNet [10], and GoogLeNet [12]. The LetNet-5 [1] is a 7-layer network structure with 32x32 grey-scale image as input for hand written digital recognition. It includes three convolutional layers (C1, C3 and C5), two sub-sampling layers marked as (S2, S4), one fully connected layers (F6), and one output layer. LeNet-5 generates a feature map and feed the feature map into the two fully-connected layers. After that, a 10-class output is generated. A receptive field (a.k.a. fixed-size patch or kernel) is chosen during the convolutional layer to compute convolution with the same size patch in the input image. A stride is defined to make sure every pixel in the original image will be covered. The system will perform convolution operation first, followed with a sub-sampling with the feature map. The goal of sub-sampling is for dimension reduction. Then, we will move to the fully connected layers, which are used to join the multi-dimension feature maps. Finally, we will generate a ten-class output, each of which represents one digital (from zero to nine). Please note, at each layer, the parameters (e.g., weight vector and bias) are trainable. Recent progresses in CNN have focused on enhancing the object representation with more complex models. For example, AlexNet [10] is a seven-layer model which includes five convolution layers and two fully connected layers. It outperformed the state-of-the-art object recognition techniques in 2012 ImageNet [65] challenges with large margin (over 10 percent). Later on, we witnessed many new models with increased layers, increased layer size, more complex neurons, as well as sophisticated computation units and layer structures. Dropout and ReLU were proposed to address the issue of overfitting and saturation, respectively. Some excellent examples include VGG net [21], ZFNet [22], GoogLeNet [12], Residual Network [66]).

The last (but not the least) related research area is edge computing service infrastructure [43, 59, 31, 39, 40, 60]. Under this infrastructure, part of the data processing tasks may be pushed to the edge of the network. One of the core ideas is called Collaborative Edge [31], which refers to the architecture that connects the edges of multiple stakeholders. These stakeholders may be geographically distributed and they may have distinct physical location and network structure. Under this infrastructure, the cloud paradigm is extended to the edge of the network. Therefore, such an edge computing service infrastructure offers a unifying paradigm for cloud-based computing and Internet of Things (IoT)- based computing. It has the potential to address the issues of delayed response time, reduced battery life, limited bandwidth, and data security and privacy. However, most of the existing use cases of edge computing-based digital health applications [41, 44] are relatively simple examples with small data sets. Novel user cases and intriguing applications with more challenging tasks, such as larger data sets and sophisticated computation, are needed for evaluating the efficacy and effectiveness of edge computing in digital health. Our proposed application, which focuses on food image recognition for dietary assessment, employ very complicated computation tasks (e.g., image pre-processing, image segmentation, and deep learning) with large image data sets (in the size of GB). This application scenario provides an excellent playground to evaluate the efficacy and effectiveness of edge computing in digital health.

4.3 System Design

4.3.1 Overview

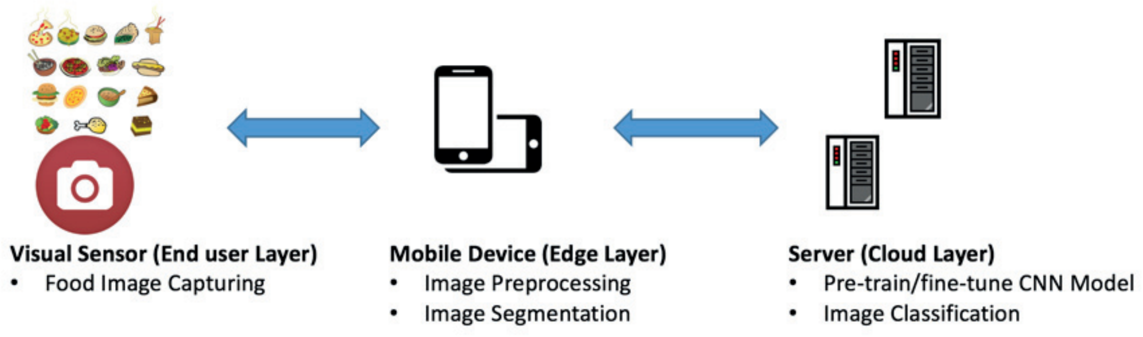


Figure 4.1: Overview architecture of "Deep Food on the Edge" system.

Our food recognition system employs visual sensors to capture food images as the source data. Due to the recent advances of electronics, visual sensors are now available in many Internet-of-Things(IoT) devices, such as smart phones. To simplify the design, we utilized the camera on smartphones for visual sensing. Besides the smartphone for sensing and image capturing, the recognition is done in a collaborative manner between the edge device (e.g., smartphone) and servers (e.g., servers in the cloud). As shown in Figure 4.1, our system includes end user layer (left most of Figure 4.1), edge layer (middle of Figure 4.1), and cloud layer (right most of Figure 4.1), together form a three-layer service delivery model. In our proposed system, data and computation are kept close to end users at the edge of network. Also, the end users device can passively record the geological location. Hence, the system could provide low latency, reduced energy consumption, and location awareness for end users. The computations are distributed throughout the network, including both the edge devices and servers in the cloud. Please note, in our system, the recognition is done in a collaborative manner.

The system design and related components are shown in Figure 4.2. As shown in this figure, our system consists of the following three major modules:

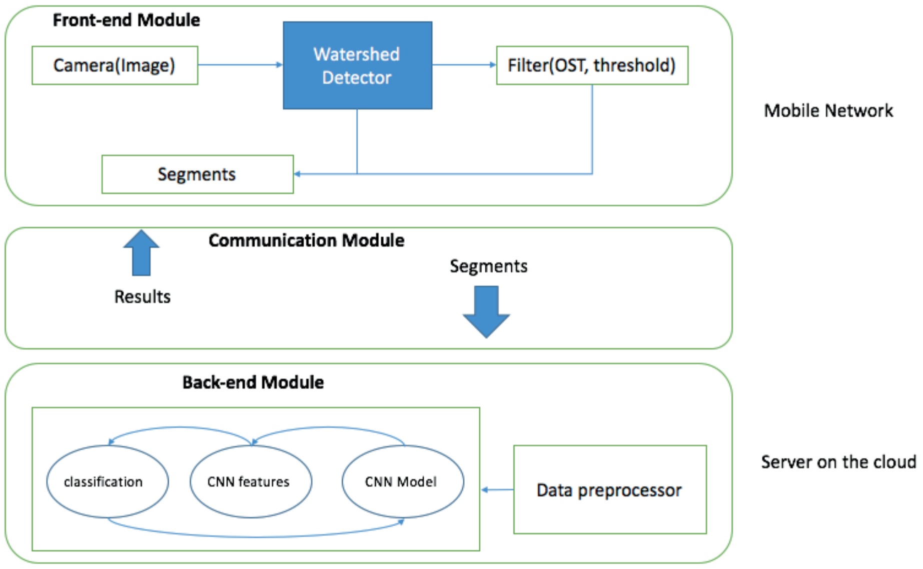


Figure 4.2: System design and components.

Front-end Component (FC). We deploy the FC module on the edge device (smartphone). As shown in the top box in Figure 4.2, its consisted of three sub-modules, which are image pre-processing (e.g., blurry image detection), watershed detectors, and the filters (OTSU or threshold)-based segmentation. After the image pre-processing module, an original clear image is generated for segmentation. Next, the watershed detector, combined with different filters (e.g., OTSU-based threshold) to segment the original image. After segmentation, we can generate the clear and segmented image. These images will be transferred to the server via the Communication Module (introduced below) for further classification.

Communication Component (CC). CC provides two channels for communication between the FC and the Back-end Component (BC), which will be introduced in more detail in the next paragraph. It transfers the image data from the FC to the BC via Input Channel, and it also passes the detection results from the BC to the FC via Output Channel.

Back-end Component (BC). The BC module runs on the cloud server, which is configured to use Caffe [67] (an open source deep learning framework) for CNN model training and testing. We use pre-trained GoogLeNet by ImageNet and fine-tune it on our food dataset (UEC-256 and Food-101). Then the trained model is deployed on the server and used for classifying the image. More specifically, the segmented image is first passed through our proposed CNN model (which is rooted from GoogLeNet model [12]), then the features are generated from the model, furthermore, a softmax classifier is used with these features to generate the probability of each category. Here we use the top-5 and top-1 probability as our prediction/classification of the food image. Our evaluation of accuracy is also based on these criteria.

4.3.2 Food Image Analysis Algorithms

In this following section, we will introduce our proposed food recognition algorithms, which runs on the FC and BC. Essentially, our system is a multiple-stage food recognition system that distributes the analytics throughout the network. This section will focus on the food image analysis algorithms part. For CNN-based algorithm, more details will be given in the following section.

Once the food images are captured, we will conduct two types of computations at mobile device in the Front-end Component (FC) (a.k.a., Edge Layer): image pre-processing and image segmentation.

The main objective of the first computation (image pre-processing) is to identify if the image being captured is blurry or not. While many cameras on mobile devices have features such as optical zoom or auto focusing, in real-world practice, when users take the pictures of food, they may have very limited time to do so due to their busy schedule and their photo taking action may be interrupted by other matters. Hence, the chances of device shaking and other interruptions while taking pictures are high. An automatic image blurry detection algorithm running at the mobile device is needed to give a real-time alarm to reminder user to re-take the picture if the image is blurry. We define an out-of-focus image as blurry image. Our goal is to develop a light weight and effective blurry image detection algorithms running at the mobile device. In literature, image restoration has been proposed to handle blurry images. Unfortunately, these existing methods are not applicable to our case because these techniques need a reference image to compute the quality of the test image. In our applications, we may only have test images. Followed our previous research [68], we propose a simple-feature(such as edginess of the image) and threshold-based method to divide the images captured into two groups (i.e., the clear image group and the blurry image group). The edginess of the image is defined as the number of edge pixels (e.g., detected by Sobel operator) divided by the total number of image pixels. The rationale behind this method is that the percentage of edge pixels for clear image (with clear object of interests) is much higher than the percentage of edge pixels for blurry image. In our previous research, we also noticed that there are different patterns between the frequency spectrums of clear image and blurry image. The Fourier spectrum of a blurry image usually shows prominent components along the certain degree (e.g., 45 degree) directions that correspond to the corners of the image. This is because the blurry image usually does not contain clear object information except the four strong edges at the corners of the image running at certain degree relative to the sides. On the other hand, the clear image usually has a lot of clear edge information so that its spectrum does not show prominent components along certain degree directions because it has a wider range of bandwidth from low to high frequencies. Based on the aforementioned observation, we first employ texture analysis algorithms on the frequency spectrum image. Then we extract different types of texture features (e.g., entropy, contrast, correlation, homogeneity) from each image. Once the features are extracted, we employ different types of classifiers to classify the images into two categories (blurry image or clear image). Similar to our previous work, we employ a two-step K-means clustering algorithms, the details is illustrated in the Algorithm 1 and Figure 4.3.

Algorithm 1 Image Preprocessing in the Front-end Component (FC)

INPUT: A set of Image $Set : \{I_1, I_2, ..., I_n\}$
OUTPUT: Two clusers $Set_b : \{b_1, b_2, ..., b_p\}$, $Set_c : \{c_1, c_2, ..., c_q\}$

```
1: $i \leftarrow 0$                                  ▷ (initialize iteration index to 0)
2: while $i$ is no more than $n$ do
3:     Read one image $I_i$
4:     Extract texture features $T_i$ from frequency spectrum
5:     Apply entropy feature extraction from $T_i$ as $S_1$
6:     Apply contrast feature extraction from $T_i$ as $S_2$
7:     Apply correlation feature extraction from $T_i$ as $S_3$
8:     Apply homogeneity feature extraction from $T_i$ as $S_4$
9:     Use binary classifier for $S_1, S_2, S_3, S_4$ separately
10:    Combine classification result using majority vote
11:    if blurry then
12:        group $I_i$ into the $Set_b$
13:    else
14:        group $I_i$ into the $Set_c$
15:    end if
16: end while
```

The main objective of the second computation (image segmentation) is to segment the image into two parts: foreground (which contains the actual food) and background. Based on the size of foreground, we could crop the image by removing

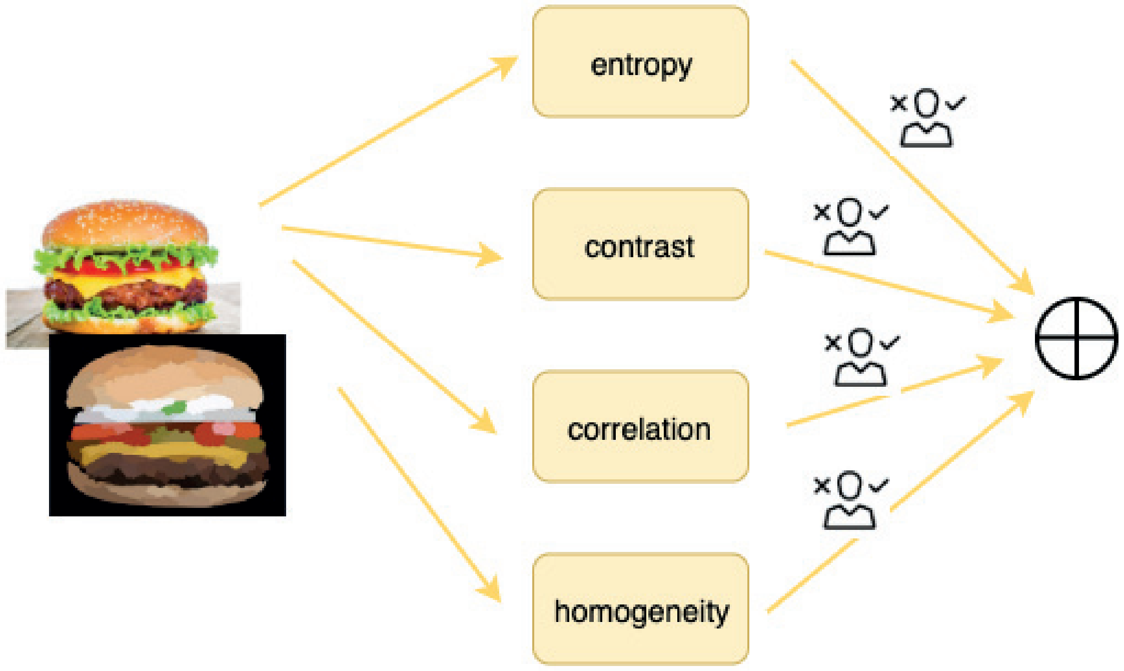


Figure 4.3: Blurry vs. clear image classification using majority vote and image features.

some portion of the background that does not overlap with foreground. According to our own experiments and other people research results, when using deep learning-based model (which is the main algorithms used in server) for image analysis and object detection, if we could reduce the background information, the object detection and recognition accuracy could be improved. Inspired by this observation, we employ watershed segmentation algorithm to preprocessing the image at FC. In this process, we first preprocess the image by image segmentation. Then we generate a new cropped image and send the updated image to the server in the cloud for further processing. By doing so, we can achieve the following performance improvements: (1) the volume of data transfer over the network may be reduced substantially. It also reduces the power consumption caused by network transferring; (2) The response time may be reduced by shorter transmission time, which will improve the user experiences; (3) The system uses much less network flow consumption, which is very helpful when the network connection is unreliable, or when the user is connected to the server via cellular network and/or he or she has limited data plan with the mobile device; (4) More importantly, the cropped image will eliminate the abundant information and further improve the accuracy for classification. In theory, the watershed algorithm is based on the following observations: any gray scale image can be viewed as a topographic surface, in which the high intensity indicates peaks and hills while low intensity represents valleys. The watershed algorithm starts filling every isolated valley with different colored water. When water rises, water from different valleys with different colors will start to merge. We could avoid this by building barriers in the locations where water merges. The algorithm continues to fill water and build barriers until all the peaks are under water. Finally, the barriers the system created are the segmentation result. Algorithm 2 illustrates the details of the algorithm.

Algorithm 2 Watershed Algorithm using topographical distance

INPUT: The lower complete grey scale Image (V, E, im), which is defined from original image with a lower boundary
OUTPUT: a sequence of labels on V, representing background or foreground

```
1:  WATERSHED ← 0                                   ▷ (The label of watershed for every pixel)
2:  Init Label with a minima and $MASK$ for other pixels
3:  U ← $\{p \in V | \exists q \in N_G(p)$: im[p] $\neq$ im[q]$\}$
4:  while not $empty(U)$ do
5:      select point $p$ from $U$ with minimal grey value
6:      remove $p$ from $U$
7:      for all $q$ steeper than $p$ do             ▷ (pixel value is greater in the neighbour)
8:          if $label[q] == MASK$ then
9:              $label[q] \leftarrow label[p]$
10:         else
11:             $label[q]$ ← WATERSHED
12:         end if
13:     end for
14: end while                                       ▷ (Label array represents the boundary)
```

4.3.3 CNN-based Food Image Analysis Algorithms

After the image pre-processing and segmentation at FC, we will further analyze these images at BC. Our proposed approach running at BC is based on the recent advances on deep learning, which aims to learn multiple levels of representation and abstraction that help infer knowledge from data such as images, videos, audio, and text.

Our proposed approach was directly inspired by CNN work from LeNet-5 [1], AlexNet [10], and GoogLeNet [12], and it employs a new module called Inception Module, which is motivated by recent advances named Network-in-Network [69]. This is also similar to the one used in GoogLeNet [12]. In this Inception Module, an additional 1x1 convolutional layers are added to the original AlexNet [10] network architecture. This additional layer undoubtedly increases the depth of the network. However, this addition could also substantially reduce the feature maps dimension. Therefore, this module could help to remove the computation bottlenecks. Specifically, we use feature map as the input for the Inception Module. After that, we apply multiple levels of convolutional layers and max-pooling layers. The kernel size of the convolutional layer varies from 1x1 to 3x3 and 5x5. At each layer, different outputs are generated and are concatenated to form the new feature map, which is used as input for the convolution and pooling operation for next layer. In order to perform dimension deduction, an optimized convolution is proposed based on the Inception Module. Please note, instead of feeding the input directly into the convolutional layer, an additional convolutional layer with size 1x1 is added to reduce the input dimension. In addition, the output from the 3x3 max-pooling layer is sent into an additional convolutional layer with size 1x1. These new designs enable the new architecture to reduced dimension even the depth of the network is increased. Not surprisingly, our experiments have demonstrated that, even under constrained computational complexity, this new network structure is able to enhance the ability to capture more visual information. Figure 4.4 illustrates the improved inception module. The network structure in the left (Figure a) is the original structure in regular CNN, such as AlexNet [10]. The right figure (Figure b) is the snapshot of the new network architecture with Inception Module. As shown in Figure b, the three added 1x1 convolutional layers are annotated with dotted rectangle and green color. While the number of layers in Figure b is four (which is one layer more than the number of layers in Figure a), the total dimension of the output (at feature concatenate layer) in Figure b is still smaller than the output dimension of Figure a.

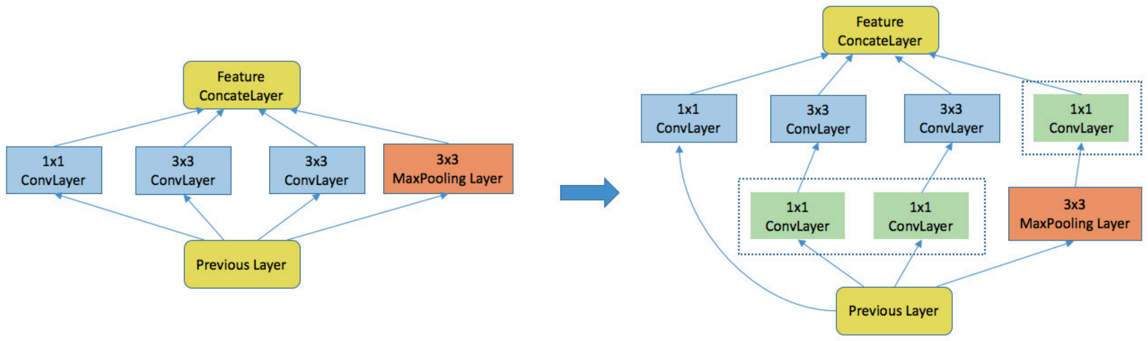


Figure 4.4: Illustration of the “Inception Module”.

The next step after forming the “Inception Module” is to employ multiple modules to form the network (similar to GoogLeNet). In this step, we will connect the two modules using one additional max pooling layer. The output from the previous module will be used as the input for the next module. Specifically, the concatenated features (output) from the previous module are fed into the newly added max pooling layer. The output from the max pooling layer is used as input for next module. Figure 4.5 illustrate this architecture. This figure includes two Inception Module, one (figure a) is located on the top of Figure 4.5 and another (figure b) is located in the bottom of the Figure 4.5. These two components (figure a and figure b) are connected via a 3x3 max-pooling layer. Essentially, the new network architecture becomes a hierarchical level step by step. In order to address the issue of increased time complexity associated with the increased network layers, we resort to the lessons learned in recent paper [70], which offer some insights for designing the network architectures by balancing factors such as depth, numbers of filters, filter sizes, etc. In this study, we design a network structure with 22 layers (similar to the one used in GoogLeNet) with different kernel size and stride. We have found that, in our study, using an input size of 224x224 with three channels (RGB), combined with 1x1, 3x3 and 5x5 convolutions, produces the best results. The 22 layers are layers with parameters. We design the pooling layer whose filter size is 5x5. The convolutional layer is 1x1 and includes 128 filters and ReLU (rectified linear activation). The dimension of the fully-connected layers is 1024. During pre-training stage, it is mapped into a 1,000-class output, similar to the ImageNet data set [65]. We use a 70% dropout rate to address the overfitting issue. Softmax is used for final classifier. Please note, based on the actual food categories, we will need to adjust the output class number during the fine-tuning stage. The proposed approach is implemented on top of open source deep learning framework Caffe [67]. CentOS 7.0 is chosen as our host system. We also use NVidia Tesla K40 GPUs for model training. The model definition is adjusted in prototxt file in Caffe. We will introduce the implementation details in Section 4.4.

4.4 System Implementation

In order to verify the efficacy and effectiveness of the proposed system, we implemented a prototype system for food recognition. Specifically, the front-end component (FC) is implemented on Android 6.0.1 (Marshmallow). The back-end component (BC) is implemented using server equipped with CentOS 7.0. The implementation of communication component (CC) includes two part. The first part is on the smartphone where we use Apache HttpClient to communicate with server. The second part is on the server we employed Django web development framework and the associated RESTful web service. In this section, we present implementation

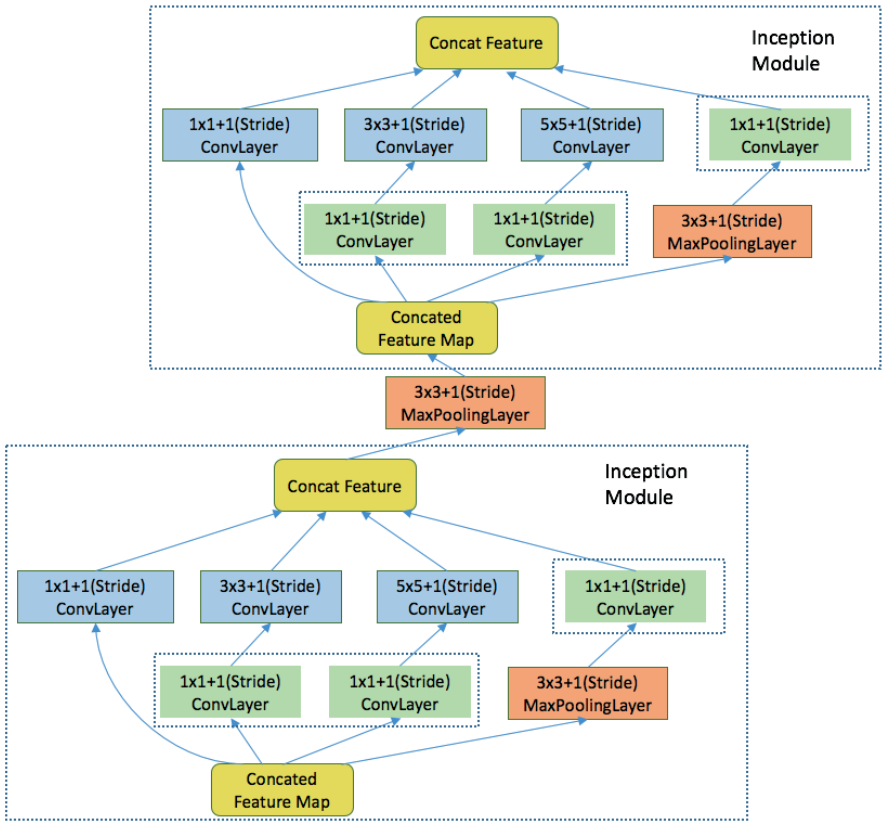


Figure 4.5: Illustration of module connection.

details.

4.4.1 Implementation of Front-end Component (FC)

We develop an Android application for the front-end module. It runs on Xiaomi Note running Android 6.0.1 marshmallow. The image pre-processing algorithm, the watershed segmentation algorithm, and the threshold filter are also implemented in this application. The watershed algorithm runs on the local mobile devices and it is implemented using OpenCV on Android devices. Several pre-defined markers are first constructed, the algorithm treats each pixel as a local topography, and then it fills the basins from the markers, until the basins meet on watershed lines. Here we set the markers as the local minimal of the food image, so that we can start from the bottom to fill the basins. We use OpenCV 3.10 and port the java SDK into the android studio project, which supports the OpenCV for Android SDK and also involves the image processing class.

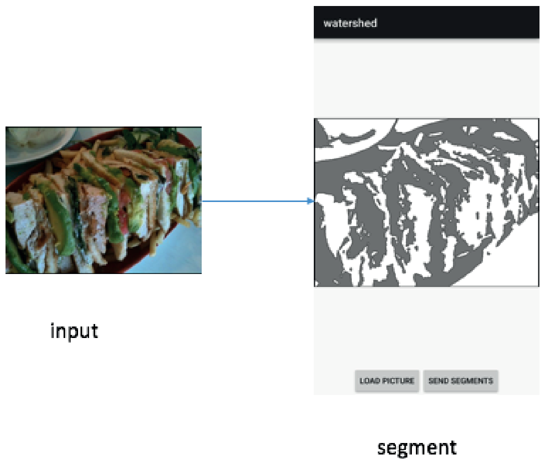


Figure 4.6: Screenshots showing image segmentation implementation in FC module.

The App we implemented has an UI for processing and loading the image. A screenshot of the UI is shown in Figure 4.6. There is a background thread for preprocessing the image. After it finishes, the App will display the segmented image in the applications mainframe. While in the background, the thread does several tasks when preprocessing the image, that includes: (1) rescaling the image if its exceed 1024x786, since too large image will increase the computing time and energy consumption; (2) converting the RGB image to grey level image for further image processing, the grey image is more easily computed when therere many channels and pixels; (3) creating the watershed class and watershed threshold for dividing the image into segments and non-segments; (4) saving and generating a unified image segments for future transferring.

4.4.2 Implementation of Communication Component (CC)

There are two implementations for communication between the Android device and cloud server. For the Android application, we use Apache HTTP Client and construct the HTTP POST request to send the segmented image into the cloud server. First, a connection bound to the server is established, and then we construct the necessary HTTP header, and fill the content with image file. Then we send the POST request to the cloud server to finish the transmission. On the cloud server, we deploy a RESTful web server using Django, which supports the file transferring (image, audio, video) using HTTP requests. When the server is up and deployed, it will listen to the port and save the requested file into the pre-configured destination. Our server will store all the necessary segments for the classification task using trained-well CNN models.

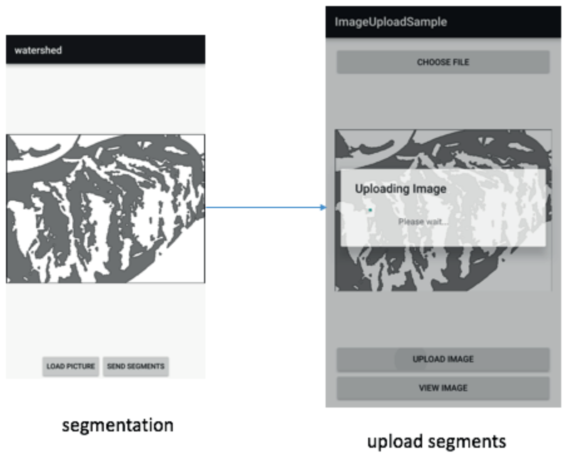


Figure 4.7: Screenshot showing segmented images being uploaded to the server in CM module.

4.4.3 Implementation of Back-end Component (BC)

Our back-end system is mainly used for classification when we receive the images from the mobile device. Before testing, we used pre-trained GoogLeNet model from ImageNet, and then fine-tuned on public food data set like Food-101 and UEC-100/UEC-256. After these steps, a fine-grained model is generated which can be used for specifically food image classification. We use Caffe to train and tune the model. And our deployment of model is also based on Caffes python interface. We first load the model into memory, when the test food image is fed into the Convolutional neural network as the input, CNN features are extracted, with max-pooling and rectified linear-unit (ReLU) layers for dimension reduction and accelerating the convergence of computing.

4.5 Performance Evaluation

4.5.1 Experiment Setup and Evaluation

In all the following experiments, we use Xiaomi Note running Android 6.0.1 Marshmallow as the front-end to install the FC of our system. This smartphone uses Qualcomm MSM8974AC Snapdragon 801 featuring Quad-core 2.5 GHz Krait 400 and an Adreno 330 GPU. It also has a 64 GB of internal storage and 3 GB of RAM. In the back-end, we use an in-house GPU server. This server is a SuperServer 4027GR-TR from SuperMicron. It has two Intel Xeon processor E5-2600 with 512GB RAM. This server is also equipped with four NVIDIA Tesla K40 GPU.

In order to evaluate the effectiveness and efficiency of our system, we implemented two other systems running the state-of-the-art visual-based food recognition algorithms for comparisons. The first one, entitled as C-System, employs different types of computer vision algorithms using hand engineered features (e.g., SIFT, SURF, HOG, Cascade) running at the mobile device for food image recognition, without relying on any algorithms running in the server. These algorithms (e.g., the Cascade algorithm) have been used in many embedded computer vision systems. We also implement the second system, called D-System, for comparisons. The D-system mainly relies on using the state-of-the-art deep learning algorithms running in the server, without using any image analysis and/or pre-processing computation at mobile device. Both systems are evaluated against our proposed system, and the performance metrics we use include response time, energy consumption, and detection accuracy.

In our experiment, we use two publicly available and challenging data sets, which are UEC-256/UEC-100 [71] and Food-101 [72]. As shown in the sub-sections below, the results of our proposed approach outperformed all the existing techniques in terms of accuracy. At the same time, the response time and energy consumption of our system are close to the minimum of the existing approaches.

4.5.2 Experimental Results on UEC-256/UEC-100

As we have introduced before, we employ two data sets for our experiments. We will introduce our experimental results for the first category in this section, which is UEC dataset [71]. It was first developed by DeepFoodCam project and the majority of the food items in UEC dataset is Asian food. This data set includes two sub-data sets: UEC-100 and UEC-256. The first sub-data set (UEC-100) includes 100 food categories with a total of 8643 images. There are around 90 images in each category. The second sub-data set (UEC-256) includes 256 categories with a total of 28375 images. There are around 110 images in each category. The researchers have added correct annotations for each image, including food category and bounding box (used to indicate the positions of the food). We use UEC-256 as the baseline dataset since we prefer to have large scale training data. We divided the images into 5 groups (5 folds). 3 groups (out of 5 groups) were used for training and the rest of the images were used for testing.

In our experiments, the publicly available, 1000-class category from ImageNet dataset was used as the pre-trained model. This model (pre-trained model) was generated by training over 1.2 million images and testing over 100,000 images. Once we have the pre-trained model, we fine-tuned this model with the UEC-256 dataset. We fine-tuned the model with a base-learning rate of 0.01, a momentum of 0.9 and 100,000 iterations. The results are shown below in Table 4.1. If we compare the results in Table 4.1 with the results in our previous publication [18], we could make two discoveries. First, our detection accuracy in this chapter is slightly better. Second, the number of iterations when we reach the best performance is less than our previous paper. These two discoveries indicate that, due to the adaption of the proposed new system and algorithms, both the accuracy and the time complexity have been slightly reduced.

# of Iterations	Top-1 accuracy	Top-5 accuracy
4,000	46.0%	77.5%
24,000	51.0%	78.8%
56,000	51.3%	79.6%
84,000	53.3%	80.6%
92,000	**54.5%**	**81.8%**

Table 4.1: Comparison of accuracy on UEC-256 at different iterations.

We also compared our results with both the C-System and the D-System. As we introduced before, the D-system is employing different sophisticated deep learning-based food image recognition algorithms, including the algorithms from the Deep-FoodCam papers. To make a fair comparison, we used the same dataset as original papers, which is UEC-100, as well as the same strategy of dividing image dataset, the result is shown in the Table 4.2. Please note, there are five C-system in this table because we tried different types of computer vision algorithms using hand engineered features. Each sub-category of C-system (the first five rows in Table 4.2) represents one type of hand engineered feature. From this table, we can tell that our proposed method outperformed all existing methods using the same dataset:

Method	**Top-1**	**Top-5**
C-System(SURF-BoF+ColorHistogram)	42.0%	68.3%
C-System(HOG Patch-FV+ColorPath-FV)	49.7%	77.6%
C-System(HOG Patch-FV+ColorPath-FV(flip))	51.9%	79.2%
C-System(MKL)	51.6%	76.8%
C-System(Extended HOG Patch-FV+ColorPath-FV(flip))	59.6%	82.9%
D-System(DeepFoodCam(ft))	72.26%	92.0%
Proposed Approach in this chapter	**77.5%**	**95.2%**

Table 4.2: Comparison of accuracy between our proposed approach and existing approaches using the same data set (UEC-100).

Table 4.3 shows the corresponding energy consumption of the three systems upon each food image. This table shows that the energy consumption of our system is very close to the energy consumption of the both C-system and D-system. Please note, in Table 4.3, we computed the energy consumption for both image analysis (on mobile device) and the image transferring (from the mobile device to the server). However, we did not compute the energy consumption if the computation is performed at the server in the cloud. Therefore, the D-systems energy consumption for image analysis is zero because D-system does not include any computation on mobile device. On the other hand, the energy consumption for image transferring for C-system is zero. Because in C-system, there is no need for data uploading since all the recognition tasks have been done on the mobile device.

Method	**Energy Consumption(Joule) for Image Analysis**	**Energy Consumption(Joule) for Image Transferring**
C-System	1.01	0
D-System	0	0.98
Proposed Approach	**0.51**	**0.57**

Table 4.3: Comparison of accuracy between our proposed approach and existing approaches using the same data set (UEC-100).

As of the computation and response time, let's first discuss the computing time. Indeed, our algorithms is based on deep learning and training a large deep learning model requires a large amount of time. For example, on a NVidia Tesla K40 GPU, it takes 2 to 3 seconds per image for forward-backward pass using our proposed architecture. Since large dataset like ImageNet and Microsoft COCO [73] contains so many images, it may not be wise to train the model from scratch. One practical strategy is to use the pre-trained model in model zoo from existing implementation (e.g., Caffe [67]), which is public for all researchers. In our own experiment, the training time is largely impacted by the computation capacity of the server (e.g., the types of CPU and GPU), how large the image candidate is, how many iterations we choose, and what value we choose for learning rate, etc. According to the rough estimation, if we use the pre-trained GoogLeNet model, then fine-tune on the UEC-100, UEC-256, Food-101 dataset, it roughly takes 2 to 3 days nonstop for a server equipped with Nvidia K40 GPU to train the model. Once the model is trained, we can directly apply the model for classifying the image. On average, it takes 50 seconds for recognition for one image. Therefore, the average response time (the time duration between capturing the image and getting the food recognition results) is 1 minute per image for our proposed approach, which include time for image pre-processing on mobile device, the time to uploading the image to server, and the time for recognition in the server. As a comparison, the response time for C-system is usually around 35 to 55 seconds (depends on what hand engineered features we use). For example, the average computation time for a SIFT-like feature extraction and analysis algorithm on a mobile device (Xiaomi Note) is 50 seconds. On the other hand, in the D-system, the response time (the time duration between capturing the image and getting the food recognition results) is 70 seconds per image in our experiments. This is mainly because in D-system, the image being processed is the raw image without pre-processing. Hence, we could conclude that the response time of our proposed approach is very close to the minimal response time of existing approach.

4.5.3 Experimental Results on Food-101

In addition to the first data set (UEC data set), we use the second data set, Food-101 data set [72], in our experiment. This dataset includes a total of 101 categories. For each food category, there are around 1000 images. We used around three-quarters (75%) of these images for training and the rest of the images are used for testing. Altogether, there are over 100,000 images in this data set. One thing about this data set is that this data set does not provide any bounding box information (which can be used to indicate the food location in the image). Instead, this data set offers food type information for each image. Different from the UEC data set, most of the images in this data set are popular western food images.

For this data set, we used a similar implementation as the one used in Section 4.4.1. The parameters were adjusted to fit for this new data set. We used a base learning rate of 0.01, a momentum of 0.9. Similar to the methods we used in Section 4.4.1, we fine-tuned the model on Food-101 dataset. Table 4.4 below shows the accuracy (both top-1 accuracy and top-5 accuracy are listed as below). Again, if we compare the results in Table 4.4 with the results in our previous publication [18], we can find that, due to the new system and algorithms in this chapter, both the accuracy and the time complexity have been slightly reduced.

# of Iterations	Top-1 accuracy	Top-5 accuracy
5,000	65.6%	88.7%
10,000	70.7%	91.2%
20,000	73.4%	92.6%
60,000	**77.0%**	**94.0%**

Table 4.4: Comparison of accuracy on Food-101 at different iterations.

We also compared our experimental results with the results of both the C-System and the D-System using the same data set (Food-101 datasets). As shown in Table 4.5, our proposed method is better than all existing work using the same dataset and division.

Method	**Top-1**	**Top-5**
C-System(RFDC-based Approach from Lukas et.al [72])	50.76%	NA
C-System(CNN-based Approach from Lukas et.al [72])	56.40%	NA
Proposed Approach	**77.0%**	**94%**

Table 4.5: Comparison of accuracy using different method on Food-101.

From the above table, we can see that pre-trained model with domain specific fine-tuning can boost the classification accuracy significantly. And fine-tuning strategy improves the accuracy comparing with non-fine-tuning method. The NA value in the top-5 column means not available, as we used the original experiment data from their paper [72], and they dont provide the top-5 result in it.

As of the energy consumption and response time, we have similar results reported as our previous data set (UEC-256/UEC-100), as introduced in the last paragraph of Section 4.5.1. Due to the space limitation, we did not report the exact numbers here.

4.5.4 The Employment of Bounding Box

As shown in both Section 4.5.1 and Section 4.5.2, the detection accuracy of our proposed approach is better than all existing approaches. We believe that one of the reasons we could achieve such performance boost is because in our proposed approach, image pre-processing and image segmentation are performed at the mobile device before analyzing these images in the server. To verify this hypothesis, we conducted a simple experiment. Our goal is to demonstrate that even very simple pre-processing can help improve the recognition performance. For example, we can use a simple bounding-box strategy to reduce the image size without analyzing the image content fully.

Specifically, we first employed the bounding box to crop the raw image. After this processing, only the food image part is remained for training and testing. Then, we performed similar experiment on UEC-256 dataset.

Method	**top-1**	**top-5**
Proposed Approach(no bounding box)	53.7%	80.7%
Proposed Approach(bounding box)	**63.6%**	**87.0%**

Table 4.6: Comparison of accuracy of proposed approach using bounding box on UEC-256.

We also conduct the experiment on UEC-100, as follows:

Method	**top-1**	**top-5**
Proposed Approach(no bounding box)	54.8%	81.4%
Proposed Approach(bounding box)	**76.3%**	**94.6%**

Table 4.7: Comparison of accuracy of proposed approach using bounding box on UEC-100.

As we can see from the two tables (Table 4.6 and Table 4.7), the employment of bounding box could boost the classification accuracy substantially. A simple explanation for this is that the abundant information in the raw image is removed after the images were cropped using bounding-box. Therefore, a more accurate and clear image candidate for training can be generated. Please note, these results are valid only if we assume the majority of food image have the foreground centered on the image. Using this simple cropping-based approach will not work well if the food is scattered on different parts of the image. In this case, our proposed approach, which conducts image pre-processing and image segmentation based on the image content, is certainly necessary to improve the recognition accuracy.

4.6 Discussion

Our findings indicated that our system achieves very high detection accuracy, as shown in previous sections. However, the response time, while very close the minimal of existing systems, is still around 5% slower than the best performer. While this is not surprising since deep learning-based algorithms are usually very time-consuming, we believe that more research should be devoted to further improving the speed. In particularly, we plan to investigate new deep learning algorithms that can be executed in mobile devices. There are some recent papers that have started to explore this area with some preliminary results [74], which further motivate us to pursue this route in the future. While pushing the deep learning-based computation further to the edge device sounds like a good idea in the initial look, we will have to consider the energy consumption if we execute the deep learning algorithms at the edge device. We believe much more research is needed in the area of distributed deep learning-based analytics in the era of edge computing.

4.7 Conclusion

In this chapter, we aimed to develop a practical deep learning based food recognition system for dietary assessment within the edge computing service infrastructure. The key technique innovation in this chapter includes: the new deep learning-based food image recognition algorithms and the proposed real-time food recognition system employing edge computing service paradigm. Our experimental results on two challenging data sets using our proposed approach have demonstrated that our system has achieved the three major objectives: (1) it outperforms the results from all existing approaches in terms of recognition accuracy; (2) it develops a real-time system whose response time is close to the minimal of existing techniques; and (3) it saves the energy by keep the energy consumption equivalent to the minimum of the existing approaches. In the future, we plan to continue improving performance of the algorithms (in terms of detection accuracy) and system (in terms of response time and energy consumption). We also plan to integrate our system into a real-world mobile devices and edge/cloud computing-based system to enhance the accuracy of current measurements of dietary caloric intake estimate. As our research is related to the biomedical field, much larger data sets are needed to provide convincing evidence to verify the efficacy and effectiveness of our proposed system. Backed by several major federal grants from NSF and NIH, we are in the process of collaborating with UMass Medical School and the University of Tennessee, College of Medicine to deploy our system in the real-world clinical practice.

Chapter 5

TUBERCULOSIS(TB) DIAGNOSIS

5.1 Introduction

Tuberculosis (TB) is a chronic and infectious disease that affects the most disadvantaged populations and involves complex treatment regimes. It remains a major public health problem with more than 9 million estimated new cases and 1.5 million deaths every year, worldwide [75]. Of the estimated 9 million people who developed TB in 2013, over 80% were in South-East Asia, Western Pacific, and African. The majority of the infected populations was from resource-poor and marginalized communities with weak healthcare infrastructure. This is unacceptable considering TB is curable and preventable. Efforts to eliminate the TB epidemic are challenged by the persistent social inequalities in health, the small number of local healthcare professionals, and the weak healthcare infrastructure found in resource-poor settings. The global health community has confronted the situation by focusing on developing and testing effective vaccines, improving the diagnosis process, and promoting patient adherence to the medical treatment.

Reducing the TB diagnosis delay is critical in mitigating disease transmission and minimizing the reproductive rate of the TB epidemic. The ultimate goal of our research is to reduce patient wait times for being diagnosed with this infectious disease by developing a socio-technical system solution to the TB diagnosis problem. Specifically, we aim to design a user-centered, mobile device-based computing system to significantly expedite the TB diagnosis process by developing novel image processing and machine learning techniques to analyze chest X-ray images. Our study will be conducted in the city of Carabayllo, a densely occupied urban community and high-burden TB area in Lima, the capital of Peru.

Mobile computing techniques offer a unique opportunity to accelerate the TB diagnosis among resource-poor, marginalized communities with weak healthcare infrastructure and systems. However, real-world mobile computing tools and applications in TB-related clinical practice with the capacity of accurate TB screening using mobile devices are rare. A wide gap between the technological advancements and the real-world clinical practices is caused by two major barriers: (1) the first barrier is the lack of large-scale, real-world, well-annotated, and public available X-ray image database dedicated for automated TB screening. For example, the majority of existing X-ray image databases, such as ImageCLEF [76], JSRT Digital Image Database [77], and ANODE Grand Challenge Database, were created mainly for one or two specific TB manifestations (e.g., pulmonary nodule). To the best of our knowledge, there is no large-scale, real-world, and public available chest X-ray dedicated for TB diagnosis with high-quality annotation; (2) the second barrier is the lack of mobile devices-based computing system that can offer accurate diagnosis by analyzing the chest X-ray images. The use of computer-aided chest radiography for TB screening and diagnosis [78, 79, 80] has been limited due to the modest sensitivity and specificity, and high inter- and intra-observer differences in reporting of radiographs. Hence, the automatic screening for TB in chest radiographs is still a challenging task and an open research problem [81]. Furthermore, there is very few reported research on using mobile device to capture and analyze the chest radio-graph images for computer-aided TB diagnosis.

Our research team, which includes computer scientists and health scientists from both U.S. and Perú, has proposed to develop a mobile device-based computing solution to overcome the aforementioned barriers. As the first step of developing such a system, we will introduce the two major progresses we recently made. The first aspect is related to the development of large-scale, real-world and well-annotated Xray image database dedicated for automated TB screening. The second aspect is focusing on developing effective and efficient computational models to classify the image into different categories of TB manifestations. Efforts described here are part of a mobile health (mHealth) integrative project aimed at reducing patient wait time to be diagnosed with TB by implementing a socio-technical solution to optimize the diagnosis process in a high-burden TB area in Lima, the capital of Perú. In this chapter, we propose a novel deep learning method with CNN and transfer learning for classifying TB manifestations in chest X-ray images. Our algorithm and training protocol show outstanding accuracy and are proven to be practical and stable for various CNN architectures(e.g., AlexNet [10], GoogLeNet [12]). Experimental results show a wide potential for medical images analysis and TB diagnosis.

5.2 Background and Related work

In this section, we will first introduce the power of mobile computing in health-care (Section 5.2.1). Then we will discuss the related work in developing chest X-ray image database (Section 5.2.2), as well as related work in computer-aided system to screen the chest radiography image for TB diagnosis (Section 5.2.3).

5.2.1 Mobile Computing in Healthcare (mHealth)

Point of care delivery is critical for the success of any application in the clinical healthcare environment. In Perú, as in many developing countries, a mobile device-based computing solution is very suitable within the context of resource-poor communities in Lima, Perú. The unique characteristics of the mobile devices such as its pervasiveness and low cost provide them the opportunity to support and enable smart care decision making in a connected health scenario for automatic health scenario for automatic tuberculosis screening.

1) **mHealth in Perú**: In a recent review of the mHealth literature published in Perú, Ruiz et al. [82] showed that mobile health interventions have enormous potential to improve access and the quality of health services in Perú, increasing the effectiveness of public health programs and reducing healthcare costs. Out of 19 papers selected, most of them showed a positive impact, and four were about tuberculosis. It is important to notice that most of them were implemented as pilot projects [82]. However, the majority of the papers demonstrated that mobile health interventions are well accepted by the population and well-developed projects might contribute to reduce the gap in public health, reducing limitations such as lack of resources (human and logistic) in heath care centers, high dispersion of the population and lack of infrastructure (roads, transportation and Internet connectivity).

2) **mHealth for TB Diagnostics**: During the last few years, mobile phones have been successfully used for diagnosis of tuberculosis [83]. In Perú, Zimic et al. [84] proposed a relatively minimal investment with mobile phones to facilitate the diagnosis of tuberculosis using a low cost Microscopic Observation Drug Susceptibility (MODS) in remote settings where a lack of trained personnel may otherwise be a limitation [84]. Nowadays, with the advances in mobile processors, images taken by a cell-phone can be immediately processed and analyzed with the help of smart algorithms. Today's global wireless infrastructure also allows transmission of a wide

variety of tuberculosis images (such as X-rays) to remote locations for telemedicine diagnosis. Therefore ubiquitous cell-phone based applications can provide unique opportunities to combat tuberculosis, especially in developing countries. Recently, Schwartz et al. [85] assessed the diagnostic accuracy of digital photographs of plain film chest X-rays obtained using a mobile phone in Botswana. The authors concluded that digital photographs of chest X-rays obtained via a mobile phone equipped with a digital camera are comparable to plain film chest X-rays.

3) **The need of a timely tuberculosis diagnosis in Perú.** In Perú, tuberculosis remains as a serious public health problem. A successful treatment plan requires a proper diagnosis, in addition to good knowledge about drug susceptibility. Reducing the tuberculosis diagnosis delay is critical in mitigating disease transmission and minimizing the reproductive rate of the tuberculosis epidemic. Different factors impact delays in tuberculosis diagnosis, such as: patient health seeking behavior, healthcare centers with poor infrastructure and equipment, inadequate resources and information systems (mostly paper-based), lack of (or in-existent) documented processes, and lack of human resources as part of a multidisciplinary tuberculosis team.

5.2.2 Developing Chest X-ray Image Database

While there are some evaluation efforts in TB screening tests on developing countries [86, 87], to the best of our knowledge, there is no large-scale, real-world, well-annotated, and public available X-ray image database dedicated for TB screening diagnosis. Most of the existing research [78, 88] in the area of computer-aided TB screening employed small data sets for evaluation and validation. Most of the datasets have less than 200 images. There are a few large data sets, such as ImageCLEF [76], JSRT Digital Image Database [77], and ANODE Grand Challenge Database, have over tens of thousands images. However, they only include one or two aspects of TB manifestations (e.g., pulmonary nodule). Without a large scale data sets with high qualify annotation, it will be very difficult to determine the efficacy of existing and proposed approach when applied to real-world clinic data. Furthermore, dedicated image annotation software tools and database storage software that can support the manipulations of the X-ray images are needed to facilitate the image annotation and image management.

5.2.3 Computer-aided System for TB Diagnosis

The research activities in the area of computer-aided image analysis for tuberculosis (TB) screening from X-ray image can be broadly divided into two categories: (1) the first category is the computer-aided screening and scoring algorithms using chest radio-graphic features for the TB diagnosis [78, 88]. Research activities in this category focus on developing different types of visual features and classification algorithms to score and screen different types of TB manifestations. Most of the papers employ texture features (e.g., Local binary patterns (LBP) [13, 14], Daubechies wavelets [15]) or geometry features (e.g., circularity, Hessian shape features). The classification algorithms employed in these papers range from simple threshold-based approach or k-nearest neighbors (K-NN) algorithm to more complicated methods, such as Decision tree and Support Vector Machine (SVM); (2) the second category of related work is focusing on X-ray image categorization on the organ and pathology level [17]. The main stream methodology in this area is based on local patch representation of the image content (e.g., visual bag of words (BoW) approach). This type of dense sampling of simple features are then feed to non-linear kernel-based classifier, such as SVM classifier. The ultimate goal is to discriminate between healthy and pathological cases. It is also shown that this type of methods can successfully identify specific pathology in a set of chest radiographs.

5.3 Proposed Approach

The ultimate goal of our research is to design and deploy a reliable, safe and secure, simple to use and power efficient mobile based cloud computing system to screen the chest radiography image with improved accuracy and reader consistency. As shown in Figure 5.1, our proposed system utilizes traditional client-server (C/S) architecture, which includes a client using mobile devices (e.g., smartphone, as shown in the left rectangle in Figure 5.1) and a remote server (e.g., server at Amazon AWS cloud computing services as shown in the right rectangle in Figure 5.1). The client and server are communicated via Wi-Fi and/or cellular network with data encrypted using Secure Shell (SSH) to ensure the security and privacy.

In this section, we will focus on the introduction of the two major progress we have made recently. As shown in the right side of Figure 5.1, the first activity is to develop a large-scale, real-world and well-annotated X-ray image database dedicated for automated TB screening. The second research activity focus on developing effective and efficient computational models to classify the image into different categories of TB manifestations. We will introduce this two progresses in the following two sub-sections.

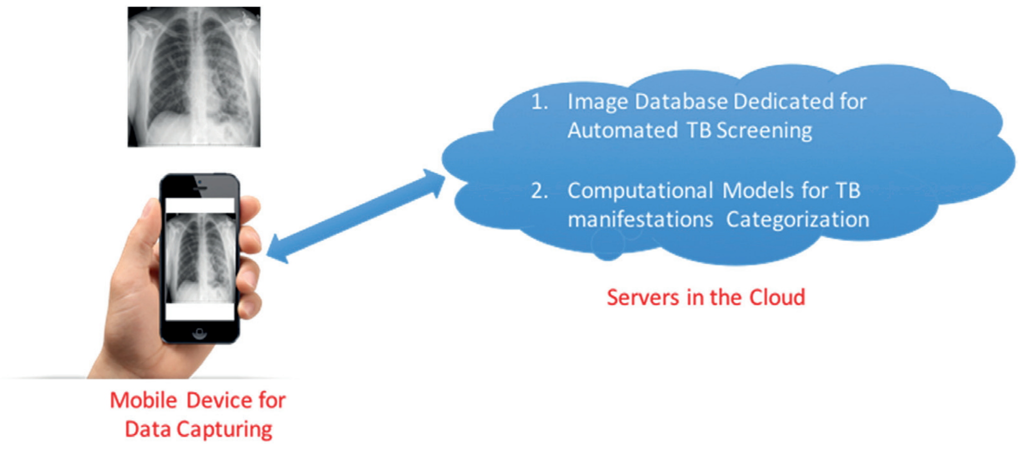

1. Image Database Dedicated for Automated TB Screening
2. Computational Models for TB manifestations Categorization
Servers in the Cloud
Mobile Device for Data Capturing

Figure 5.1: Overview of proposed mobile based system for improving TB diagnosis.

5.3.1 Developing Chest X-ray Image Dataset

To first build deep learning model, we have to build a large-scale, well-annotated chest X-ray image dataset, as there is no such big dataset available. The main challenge in this component includes: (1) where and how to gain access to the real-world, large-scale TB screening images with detailed diagnostic descriptions; (2) to determine the types of TB manifestations we should target and how we can use these manifestations to annotate each X-ray images; and (3) to develop dedicated annotation software and database management software package for reviewing the chest radiography, locating important contents, annotate them, and extract the annotated contents for research, teaching, and training purposes.

Here we propose several approaches to solve the above issues. To address the first challenge, we established an international research team which include scientists from both U.S. and Perú. One of the core team members is Dr. Jesus Peinado, header of Informatics at Partners In Health at Perú. In the past three years, his team in Perú has collected around 5,000 chest X-ray radiography images captured from real-world TB patients with detailed TB screening descriptions. In addition to that, we also explored the second source of image is the X-ray images from the 2004 - 2013 ImageCLEF collection, which include over 400,000 medical images, diagnostic annotations, search topics and relevance judgments. The 2004 - 2007 collection [89, 90, 91] contains over 66,000 images from a variety of teaching files annotated in English, French or German. The 2008 - 2010 collection [92, 93] contains over 77,000 images and captions from the medical literature. These images were published in Radiology and Radiographics, two of the journals published by the Radiological Society of North America. The 2011 - 2013 collection [94] includes more than 300,000 image and related text annotation from the biomedical literature (e.g., PubMed). To address the TB manifestation issue, we worked very closely with our clinical and research collaborators, Dr. Jesus Peinado from Peru and Dr. John Bernardo at Boston Medical Center(BMC) and Boston University School of Medicine (BUSM), to generate a scientific categorization of TB manifestations. As shown in Figure 5.2 we have identified five TB manifestations. There are some important discoveries (which will serve as important motivations and rational for the proposed approach for image analysis and machine learning techniques) from these images. First, the variety of the TB manifestations is large. Second, each category of TB manifestation actually indicates the severity of the TB disease. Therefore, algorithms that can recognize and classify the X-ray image into different type of TB manifestation can serve the purpose of screen the X-ray image to understanding the severity of the TB disease.

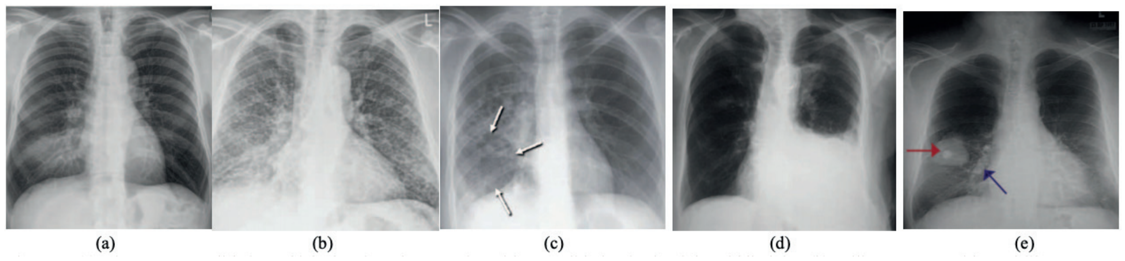
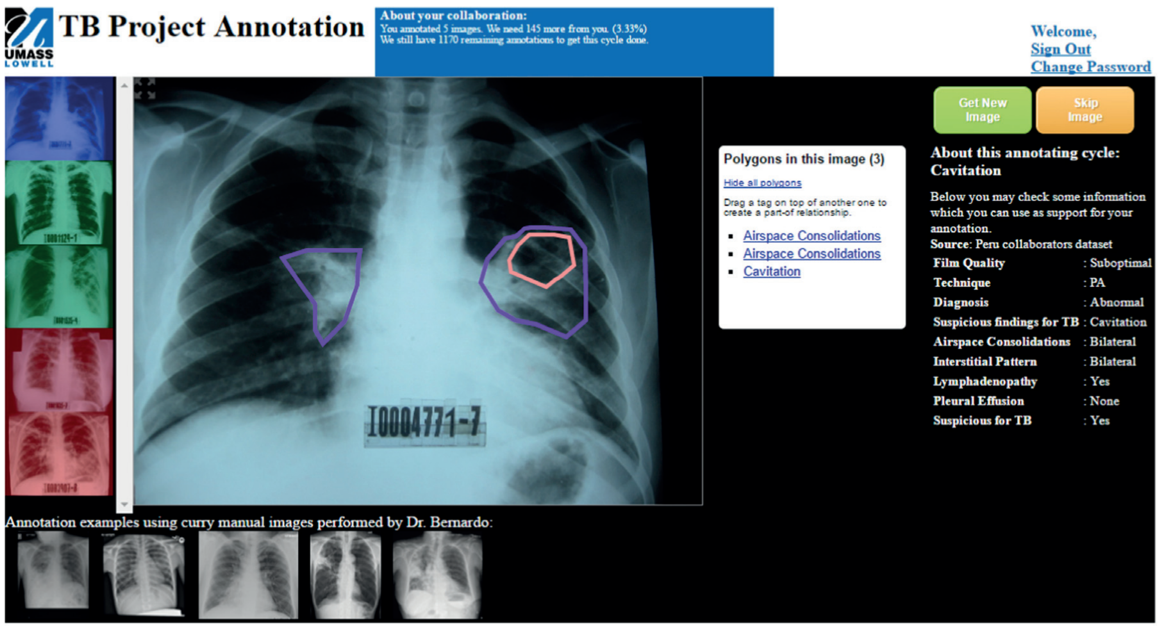


Figure 5.2: (a) Air space consolidation which showing glass opacity with consolidation in the right middle lobe; (b) Miliary pattern with seed-like appearance; (c) Cavity located at the lower lobe (annotated by arrows); (d) Pleural effusion, which is excess fluid that accumulates in the pleural cavity; (e) Calcified granulomata: The red arrow indicates a large 5 cm diameter squamous cell carcinoma of the right lower lobe and there is 1.5 cm bright opacity in the middle of the mass (which is a calcified granuloma). Additional calcified granulomatous areas are medial to the mass, as indicated by blue arrow.

5.3.2 Collecting Chest X-ray Image Annotation

In addition to developing the real-world database, we also developed annotation software for reviewing the chest radiography, locating important contents, annotate them, and extract the annotated contents for research, teaching, and training purposes. While there are many existing efforts in medical image annotations [95, 96, 97], there is few open source annotation software dedicated to annotating X-ray image to support automatic screening. Based several existing open source annotation software projects [98], we developed a web-based annotation software.

In general lines, the annotation software is a tool in which someone capable can highlight the location at the image in which some TB manifestations is occurring. However, this task is not possible without the guidance of a specialist. We trained a team with around 10 members who were trained directly by the Pulmonologist Dr. John J. Bernardo. The train were provided for each TB manifestation isolated, and some extra training to verify complex cases. The annotation is performed using our own software developed exclusively for this purpose. The common interface of the annotation software can be seen on Figure 5.3. In the left panel are all the images in which the annotator is working on.



About your collaboration:
You annotated 5 images. We need 145 more from you. (3.33%)
We still have 1170 remaining annotations to get this cycle done.

Polygons in this image (3)

Drag a tag on top of another one to create a part-of relationship.

About this annotating cycle: Cavitation

Below you may check some information which you can use as support for your annotation.
Source: Peru collaborators dataset

Film Quality	: Suboptimal
Technique	: PA
Diagnosis	: Abnormal
Suspicious findings for TB	: Cavitation
Airspace Consolidations	: Bilateral
Interstitial Pattern	: Bilateral
Lymphadenopathy	: Yes
Pleural Effusion	: None
Suspicious for TB	: Yes

Annotation examples using curry manual images performed by Dr. Bernardo:

Figure 5.3: Annotation software interface.

The images have a color to identify the current annotation status: blue for the image shown in the central panel, green for the images already annotated and red for images with no annotation. On the middle, we have the main panel which contains the current image with every annotation on it, the system pick one specific color for each different manifestation, on the right side of the annotation panel the annotator can see all the annotated polygons listed, also, the annotator can hide the polygons to see a clean image and can highlight individually each polygon as well. When the annotator starts an annotation, he/she clicks in a point to be the initial vertex of a polygon, after, he/she keeps indicating each subsequent vertex of the polygon, and finally close the polygon with a click in the initial vertex, in the moment in that the polygon is finished, a popup window shows up asking the details about the regions highlighted, it includes the TB manifestation, the confidence level, and possibly some notes (Figure 5.4 - left). The manifestation may be informed using some default options in a select box (Figure 5.4 - right), or the annotators may choose to pick the option other and write the manifestation themselves.

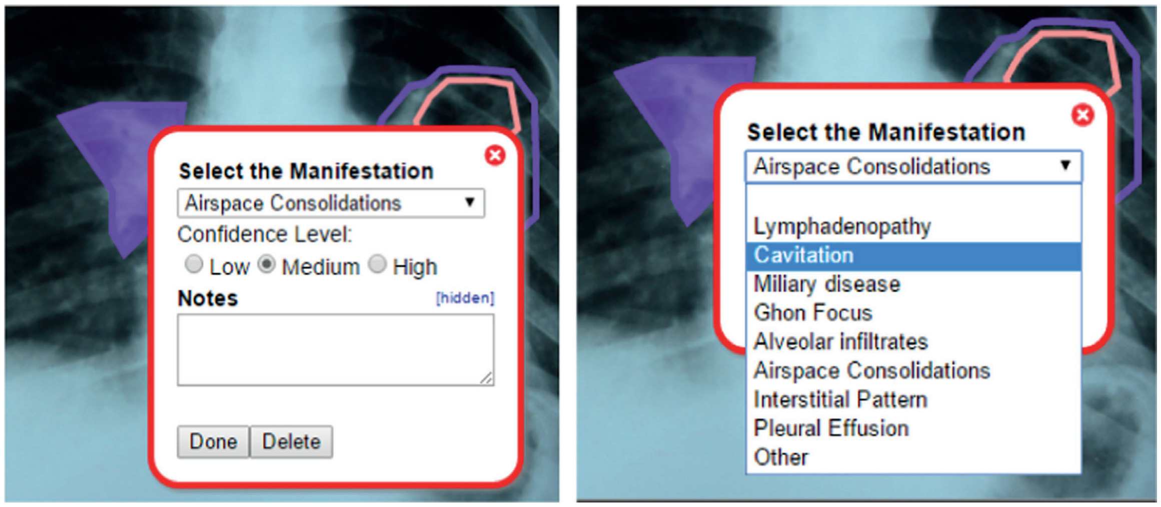
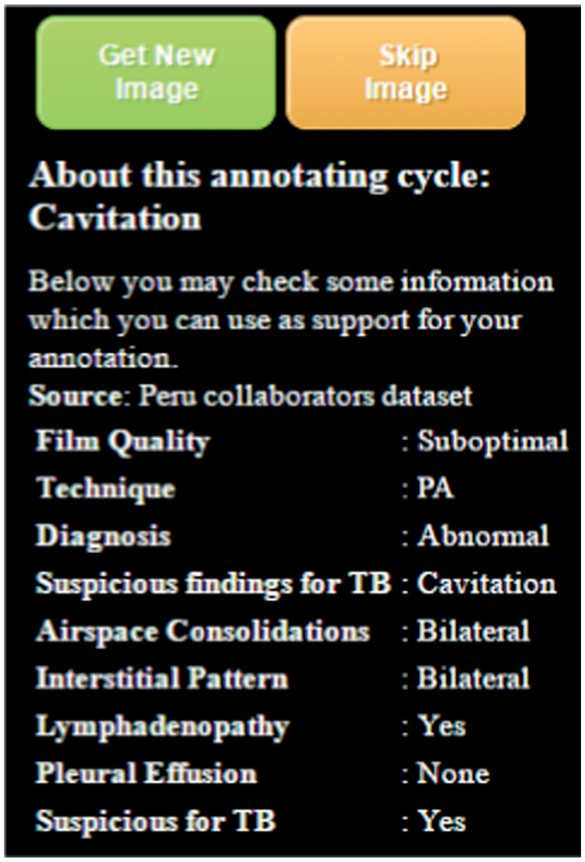

Select the Manifestation: Airspace Consolidations
Confidence Level: Low [✓ Medium] High
Notes [hidden]

Figure 5.4: Pop up to inform the manifestation and details.

The right panel (Figure 5.5) shows important information provided by the team in Perú coordinated by Dr. Peinado that helps the annotator to provide the correct manifestation, the items cover the main TB manifestations possible to be found in x-ray images (seen in Figure 5.5) may be verified with some details in this panel, including sometimes the side (right, left or bilateral) in which a specific manifestation is. Also, the right panel provide two buttons, the first one is a “Get New Image” that adds to the annotator’s personal collaboration one more image to be annotated. This button, randomly select an image lacking annotation and gives it to the annotator, so, there is no pre-selected image to a specific annotator. The annotator also may skip the image using the button “Skip Image”, it makes the current image returns to the set of non-annotated images, any annotation made is discarded and the image are available to any annotator again.



About this annotating cycle: Cavitation

Below you may check some information which you can use as support for your annotation.
Source: Peru collaborators dataset

Film Quality	: Suboptimal
Technique	: PA
Diagnosis	: Abnormal
Suspicious findings for TB	: Cavitation
Airspace Consolidations	: Bilateral
Interstitial Pattern	: Bilateral
Lymphadenopathy	: Yes
Pleural Effusion	: None
Suspicious for TB	: Yes

Figure 5.5: Right Panel, buttons and golden information.

The information shown in this panel is not the only support the annotators have, on the bottom there are some images annotated by a pulmonologist using the same software to work as a sample to the annotators, when any image is clicked, a new window opensshowing the sample as in Figure 5.6.

As mentioned, the annotation software is designed for capable annotators, in our context, capable annotators are people who received a training to identify and localize TB manifestations. To make annotations in an x-ray is a tough task even to experienced pulmonologists. Because of this, the annotators received one training for each TB manifestation, during the training also is collected some images to be used as a sample as shown in Figure 5.6, after the training all the annotators work in the same manifestation until all images from that specific manifestation is covered. However, images may contain more than one manifestation at the same time, in these cases, only the current manifestation is being annotated, and the remaining wait for the next training.

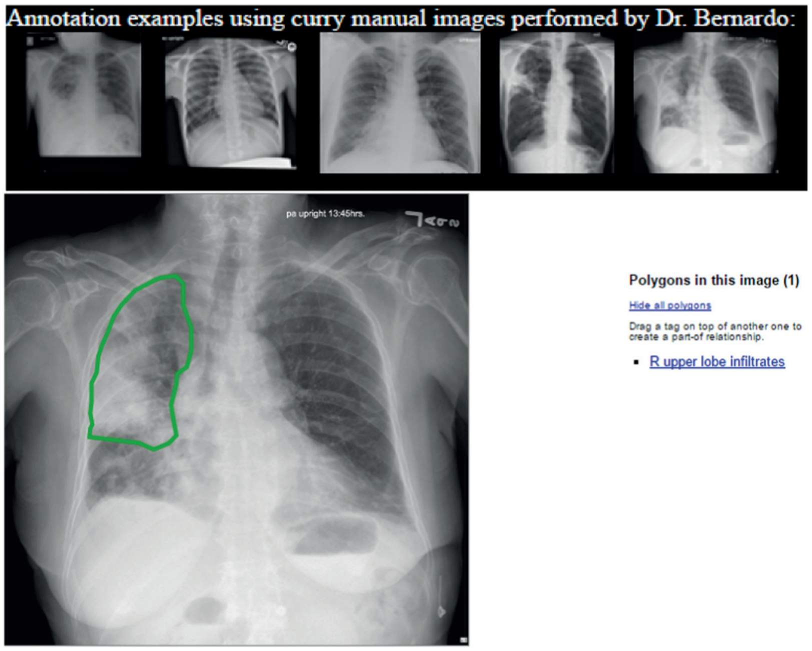

Annotation examples using curry manual images performed by Dr. Bernardo:
pa upright 13:45hrs.
Polygons in this image (1)
Hide all polygons
Drag a tag on top of another one to create a part-of relationship.
R upper lobe infiltrates

Figure 5.6: Images annotated by a pulmonologist (top) to be choose, one specific image open as a sample (bottom).

We call the training and the following annotation as cycle as we can see in Figure 5.7, the cycle repeats for each new manifestation. In each cycle the annotation software also provides a feedback to the annotator about your current collaboration, including how many images left in your work or in the entire cycle. The cycles starts with the training following to the annotation and, after finished the annotation for each collaborator, a new meeting with the pulmonologist is made to work on cases annotated with low confidence, for this review the pulmonologist has a special page that loads only the images with low confidence and without any button, the pulmonologist can edit the annotation or just change the confidence level if it is well annotated, the pulmonologist has access to the collaborator responsible by that one to further clarification.

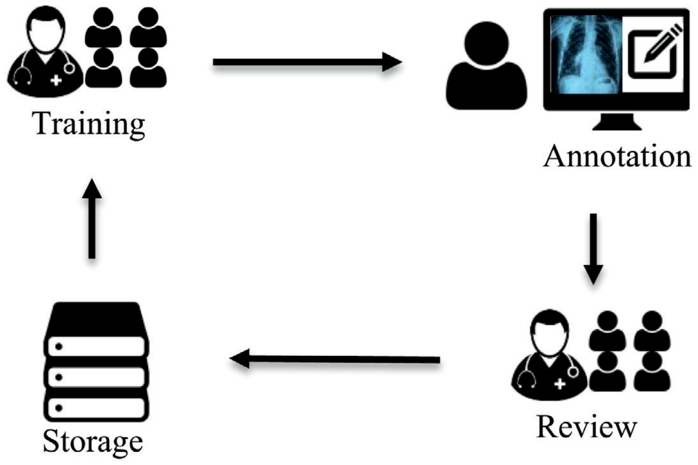


Figure 5.7: Cycle for annotation process that repeats for each TB manifestation.

After this closing meeting, all the annotatios are stored and no more are going to appear to annotators. From the second cycleonwards, the same images may appear again asking for the annotation of different manifestations, in these cases, all the previous annotations are shown to help the annotator to differentiate the current manifestation from the previous ones, the annotator also can correct the previous annotation for a more accurate one.

After all the cycles, except by the images containing health lungs, all the other images contain the annotations for each manifestation so the images are ready for be used in a region based classifier.

5.3.3 Convolutional Neural Network

After collecting the large-scale, well-annotated chest x-ray image dataset, the next step is to research and develop effective and efficient computational model for TB manifestation analysis. There are several major challenges for solving this problem. The main challenges of automatic TB screening come from the extremely complexity and large variety of the TB manifestations. This is true in clinic practice. As we have shown in Figure 5.2, the variations of TB manifestations can range from subtle military patters to apparent effusions. Via close collaborations and discussions with domain experts, we have discovered that unusual or abnormal TB manifestations affect the texture and geometry of the anatomy. Therefore, most of the existing techniques employ texture and/or geometry features. Usually, different features are useful for different manifestation. For example, texture features, such as Mean, Variance, Entropy, and Third moment, can be employed for detecting infiltration and dense. Local binary patterns (LBP), another type of texture spectrum feature, can be used for cavity detection. Template matching on Fourier domain, a method for geometry feature extraction, maybe useful for detecting the miliary pattern. Hessian shape features, another type of geometry features, could help to detect the nodules. Recently, researchers [78, 79] have shown that combine multiple features can improve the performance of abnormal TB image detection. For example, in paper [79], LBP and histogram of oriented gradients (HOG) are combined for cavity detection. In paper [78], a mixture of Intensity, LBP, and Hessian shape features are employed to measure normal and abnormal patterns in the X-ray image. The literatures have shown that the choice of features play the key role for system performance. Hence, the key issue is how to improve these hand-tuned features. To address these challenges, we plan to explore new solutions based on recent advances in deep learning [99, 100, 97].

Deep learning, aims to learn multiple levels of representation and abstraction that help infer knowledge from data such as images, videos, audio, and text, is making astonishing gains in computer vision, speech recognition, multimedia analysis, and drug designing [18]. Briefly speaking, there are two main classes of deep learning techniques: purely supervised learning algorithms (e.g., Deep Convolutional Network), unsupervised and semi-supervised learning algorithms (e.g., Denoising Autoencoders [19], Restricted Boltzmann Machines, and Deep Boltzmann Machines [20]). With the help of large-scale and well-annotated dataset like ImageNet, its now feasible to perform large scale supervised learning using Convolutional Neural Network(CNN). The issue of convergence has been addressed by Hintons work in 2006. Subsequent theoretical proof and experimental results both shows that large scale pre-trained models in large domain, with specific small scale unlabeled data in another domain, will give excellent result in image recognition and object detection. To address the issue of limited abilities of feature representation, many researchers have proposed more complex CNN network structure, like VGG [21], ZFNet [22], GoogLeNet [12] and so on. On the other hand, ReLU [23] is also proposed to make it converge faster and also gains a better accuracy. Most of current researchers have put efforts in making the network deeper and avoid saturation problem.

Convolutional Neural Network(CNN), on the other hand, is widely used in multiple computer vision tasks, such as image classification, object detection and visual question answering. While many efforts have been given to general tasks, few of them focus on medical images. Here we study two convolutional neural network architectures(AlexNet [10] and GoogLeNet [12]) with different model parameters. Comparing with general image applications [10, 12], deep learning models for medical images tend to be smaller [101], as the region-of-interest(ROI) are usually small. We adopt this schema and use various CNN models and smaller size of kernels to find the best fit for the TB chest X-ray images.

Figure 5.8 shows a basic structure for TB classification using LeNet [1]. A CNN model is usually consist of convolutional layers, pooling layers, and fully-connected layers. Each layer is connected to the previous layer via kernels that have predefined, fixed-size receptive field. The weights within each layer are shared to reduce complexity and computation. CNN model learns the parameters from a large-scale dataset to represent the global and local features in the image. Every model architecture has various types of layers and activation functions to exhibit strong feature representation ability than human-engineered features. More details of the network structure are discussed in [10, 12, 57, 97].

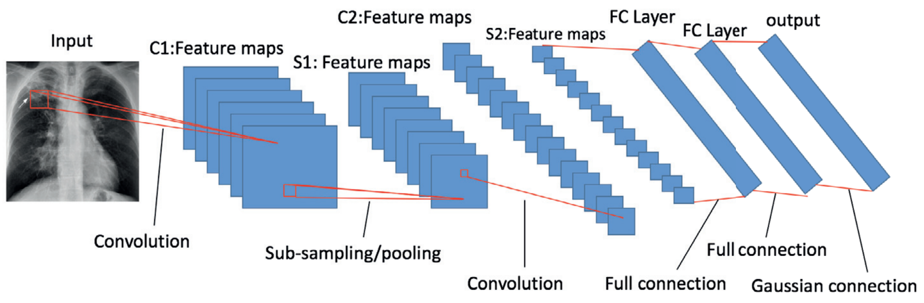


Figure 5.8: CNN architecture(LeNet[1]) for TB classification.

5.3.4 Transfer Learning

Transfer learning aims to store the learned knowledge from one domain and apply it to another different but related domain. When training from scratch, it usually takes a lot of time because model parameters are all initialized with random Gaussian distribution and convergence are achieved after at least 30 epochs with a batch size of 50 images. Another challenge is that in the medical domain it's usually very hard to obtain large-scale, well-annotated images. Lacking of medical data usually makes it very hard to learn precise models for accurate predictions.

Recent studies[45, 102, 42, 103, 97] show that using a pretrained model from ImageNet dataset, then finetune with a more specific dataset yield outstanding classification and detection results. The reason behind this training protocol is that CNN gains general representation capability from pretraining in natural images. After finetuning, the model adjusts the parameter for representing the unique features in the specific images, while retaining the abilities to represent general image. We inherently adopt this training strategy, combine with shuffle sampling and cross-validation, and creatively apply it to the chest X-ray images for classifying TB manifestations. Our experiment shows the outstanding performance for TB diagnosis.

5.3.5 Detecting TB manifestation

Our main objective is to analyse the X-ray images and to screen the chest radiography image with improved accuracy and reader consistency. We convert the screening problem into a classification problem. More specifically, we will investigate effective and efficient computational models to segment the image into smaller region and classify the image region into different category of TB manifestations (e.g., Air space consolidation, Miliary pattern, Cavity, Bronchiectasis, Opaque, etc.). As shown in Figure 5.9, our proposed approach includes the following three steps.

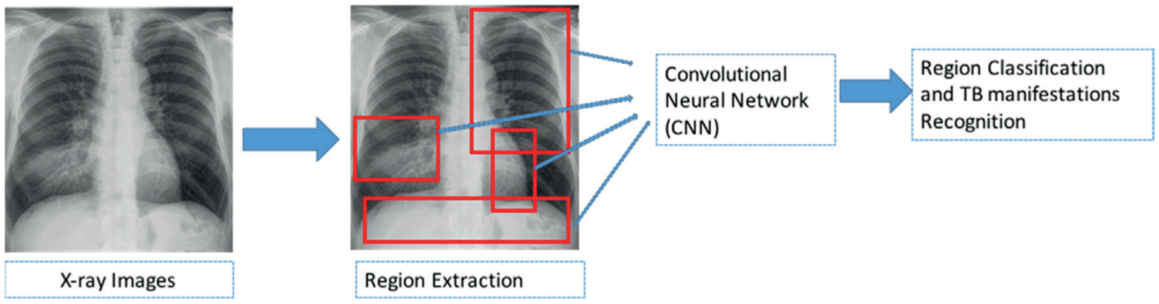

X-ray Images
Region Extraction
Convolutional Neural Network (CNN)
Region Classification and TB manifestations Recognition

Figure 5.9: Proposed approach for X-ray image analytics.

1. *Extraction of region proposals*: In this step, we will extract regions from the image using different methods like selective search [104]. For each region, train a CNN model to calculate the new features for further classification. Please note, before we perform the feature extraction process, the region should be scaled to a fixed size 227x227 (in order to the same vector dimension in our further handling). After the above handling, we should generate a 4096-dimensional feature vector. The features from CNN will be combined with features originally transmitted from mobile phone. We will apply the linear classifier like Support Vector Machine (SVM) [105], the combined features for final region classification and TB manifestations recognition. For the implementation purpose, we use the open source Caffe [67] for training. The training of the proposed CNN approach could include two steps: (a) supervised training on a large dataset using CNN;

(b) fine-tuning the CNN feature for detection using a smaller dataset. Our preliminary study has shown the feasibility of the proposed approach for a small group of images. In this study, we plan to extend and refine our preliminary results to large scale, real-word X-ray image datasets.

2. *Initial Image feature extraction*: In this step, we will extract both global and local features from the X-ray images captured by the mobile device. While there exists a large number of global features ranging from color, texture, to edge features, we mainly choose texture and shape features because the cutting-edge research in computer-aided TB screening using X-ray imaging have shown that texture and shape features are most effective. Some sample global features we used include: Gabor features and Local Binary Patterns (LBP) features. Some sample local features include SIFT features [4] and PHOG [106].

3. *Deep Convolutional Neural Network (CNN)-based X-ray Image Analysis*: Recall in our first step, we have extracted some features from the original image. Hence, we do not need to transmit the entire image. Instead, we resize the image at mobile phone and only transmit the image with much smaller size. In our preliminary test, we reduce the size by half and the results are still acceptable. By doing so, our system will consume substantially less power, compared with transmitting the original image. Another contribution in this component is that we plan to employ the deep convolutional neural networks (CNN) for region classification. Our proposed techniques are rooted from recent advances on deep learning, such as region deep convolutional neural networks [11, 107], which take full use of region features.

5.4 Experiments

We conducted several experiments on a private TB X-ray image dataset from Perú and followed the standard evaluation protocol, used multiple training models and got the average as the final accuracy.

5.4.1 Dataset Details

Category (Name of TB Manifestation)	Total Image
Miliary Disease (MI)	25
Cavitation (CA)	1182
Lympahadenopathy (LI)	202
Ghon Focus (GH)	27
Alveolar Infiltrates (AI)	2252
Other (OT)	560

Table 5.1: Data distribution in TB dataset.

The dataset is from our Peruvian partners at "Socios en Salud", Partners In Health in Lima, Perú. This dataset contains 4701 images, 453 of them are labeled as normal (which means the patients don't have the TB) and 4248 are labeled as abnormal that contain various TB manifestations. Among the abnormal TB images, there are 6 categories, which indicate 6 different types of TB manifestations: miliary pattern, cavitation, lymphadenopathy, ghon focus, alveolar infiltrates and others. Table 5.1 illustrates the characteristics of the data. This less-category and imbalanced distribution casts a unique technical problem for classifying the image. In the next sections, we will show our architecture design and optimization that improves the performance by a large margin.

5.4.2 Architecture Details

We revise the AlexNet and GoogLeNet architecture for image classification. On the top of AlexNet/GoogLeNet, we place a softmax layer to get the score for each category. Each score ranges from 0 to 1 and represents the probability of classifying the manifestation correctly. When training AlexNet, we use the standard 7-layer network structure[10], which is consist of five convolutional layers and two fully-connected layers. Dropout and ReLU are deployed to address overfitting and convergence issue. For GoogLeNet, we use the 22-layer network structure[12] that employ the Inception model with dropout and ReLU. We implement our architecture using Caffe. Our model is trained using Stochastic Gradient Descent(SGD), with a lot of combinations of different parameters. By experiment, we find that for AlexNet, a base learning rate of 0.01, a momentum of 0.9 and weight decay of 0.0005, would yield the best result within reasonable time. For GoogLeNet, we use a base learning rate of 0.001, a momentum of 0.9 and weight decay of 0.0002. Note that learning rate is not fixed and will decrease exponentially as iteration grows. Since the chest X-ray images are still not abundant for training large complex model, we use the pretrained model from ImageNet that are public available in Model Zoo among the Caffe community and then finetune on our chest X-ray image database.

5.4.3 Shuffle Sampling

For very imbalanced dataset, it's very hard to learn a general classifier for all categories. How to augment data is crucial to learn an expressive classifier to classify all TB manifestations. Inspired by the previous work from [108], we propose a shuffle sampling technique to augment data. Our method is done before the training of CNN models, so it doesn't affect the training time substantially. According to our experiments, the training time for using shuffle sampling increases about 2 hours for our 5K images with AlexNet, while the accuracy boost from 53.02%[42] to 85.68%.

As Figure 5.10 shows, we first select the largest number of category as the baseline number of instances, marked as N. Then for each category, we generate N *unique* integers, each integer represents the index for one image. For each category,

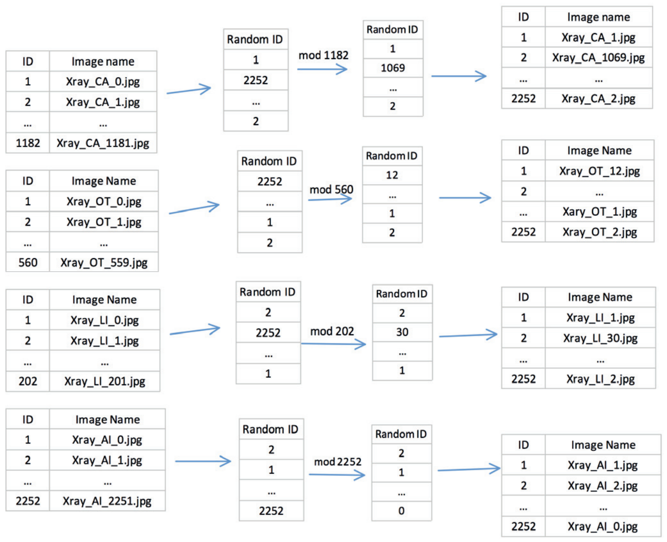


Figure 5.10: Shuffle sampling for imbalanced data.

we calculate the *mod* value $I_i\%N_c$ as the final index, I_i is the index from the unique sequence of each category, N_{ci} is the number of maximum images for each category. Here N is 2252, and I_i is from the random array. After these shuffle samplings, we expand our dataset into $N * c$, where N is the maximum number of images for all the categories, and c is the number of categories.

5.4.4 Results

We study several cases of classifying the manifestations and evaluate their performance. For each model, we use non-shuffle and shuffle sampling settings for four common manifestations, these manifestations have more training images than others. We first pretrain on AlexNet using ImageNet dataset, then finetune using the same network structure with chest X-ray images. Figure 5.11 shows that shuffle one is more accurate than non-shuffle one in all iterations and remains stable in general cases.

The final classification accuracy using AlexNet is about 85.68%, a significant improvement from non-shuffle sampling's 53.02% in[42]. Note that since this strategy may generate repeated images for testing, influencing the final evaluation of accuracy,

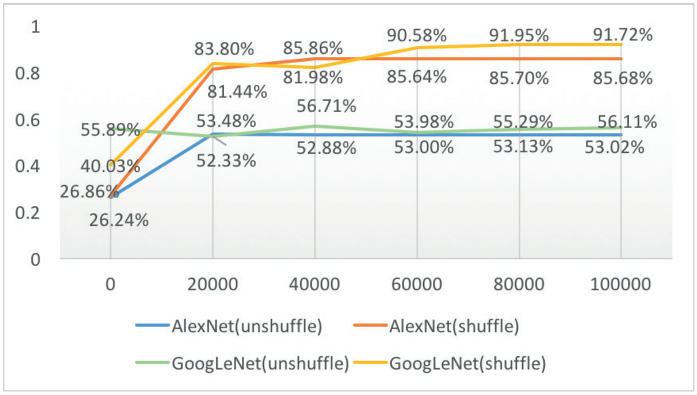


Figure 5.11: Non-shuffle vs. shuffle classification accuracy.

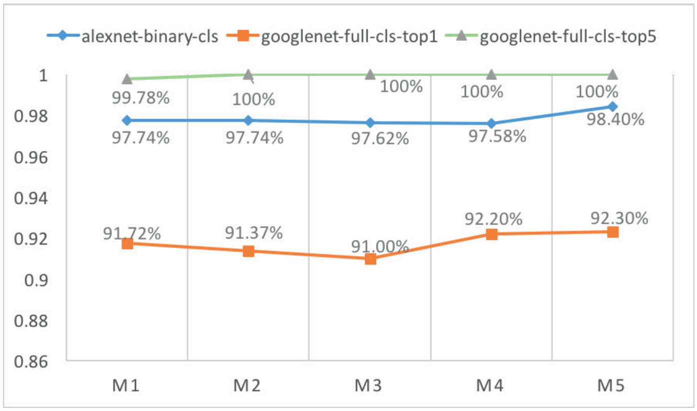


Figure 5.12: Classification accuracy with cross validation. Use AlexNet for binary classification and GoogLeNet for full classification

we use the trained-well model to retest all the images in the training data with the unique id to verify its correctness, we calculate the precision, recall, f1-score and miss rate. Table 5.2 shows the stability of our method to classify the images on the original training set.

We also conduct an experiment on abnormal detection using AlexNet, which is to determine whether a chest X-ray image contains TB or not. This is a binary classification problem, we use cross-validation repeatedly. By dividing the whole dataset into 5 equal folders, we used 4 folders as the training set and the remaining 1 folder as the test set. We also explore the deployment of GoogLeNet for all six manifestations with shuffle sampling and cross-validation to enhance the model prediction. Results are shown in Figure 5.12 and Figure 5.13.

Set	Class name	Precision	Recall	F1-score	Miss rate
Train	CA	0.90	0.94	0.92	0.06
	OT	0.87	0.99	0.93	0.01
	LI	0.91	1.00	0.95	0
	AI	0.98	0.91	0.94	0.09
Test	CA	0.87	0.93	0.90	0.07
	OT	0.90	1.00	0.95	0
	LI	0.91	0.98	0.94	0.02
	AI	0.97	0.90	0.93	0.10

Table 5.2: Evaluation on training and testing set

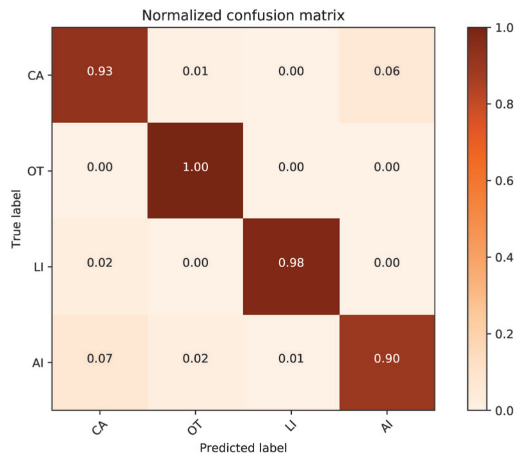


Figure 5.13: Confusion matrix on test set

Finally we evaluate the computation time. We use a Telsa K80 GPU for training and testing, it roughly takes 2 to 3 days nonstop to train the AlexNet or GoogLeNet models. After we have trained the model, we use pycaffe, scikit-learn and pandas-ml to evaluate model performance, it takes less than 1 minute to test one image when using GPU.

5.5 Conclusions and Future work

We design a novel method to apply CNN models to detect and classify TB manifestations in X-ray images. Work presented here is the first research trial using CNN for TB detection in a large TB dataset. Based on the research result and the specific technical problems in this large imbalanced, less-category dataset, we use a set of optimization solutions to further improve the accuracy. Our method shows the stability and universality in various CNN architectures. The next step is to collaborate with health science and engineering researchers to annotate the regions of the chest images for more accurate classification and localization. We will use more region-level information for preprocessing and study the algorithms to further improve the accuracy. We will also deploy a user-centered, mobile device-based computing system to expedite the TB diagnosis process and conduct the field-testing in TB clinics in a high-burden TB area in Lima, the capital of Perú.

Chapter 6

VEGETABLE IMAGE RECOGNITION

In self-service supermarket and retail industry, efforts to reduce the wait time for customers using automatic identification of grocery items are challenged by low recognition accuracy, long response time and substantial requirement for equipment. In this chapter, we propose a novel edge computing system named EdgeVegfru for vegetable and fruit image classification. While existing work on Vegfru dataset shows outstanding performance, few of them have been deployed in real-world applications. We adopt an edge computing paradigm and implement the whole system on the Android devices. The proposed deep learning model and quantization algorithm reduce the model size and inference time significantly. Our system has shown outstanding accuracy within limited time and computation capability, comparing with other machine learning methods(Support Vector Machine(SVM), Gradient Boosted Tree(GBT) and Random Forest(RF)), thus providing the potential path for automatic recognition and pricing in self-service retail stores.

6.1 Introduction

Self-service Technologies (SSTs) are those interfaces that allow customers to experience service without directly dealing with the service employees [109]. Emerging technologies in artificial intelligence and computer vision are adopted in self-service facilities to save labor cost and improve the check-out efficiency. However, previous advances in technology is still lagging behind under the strict accuracy and response time constrains. In the retail industry, especially the grocery stores and food markets, the need for high-accurate, low response-time intelligent system is becoming a critical factor to improve customer satisfaction and store profits. In this chapter, we focus on developing efficient and accurate algorithm and system for automatic recognition of vegetables and fruits for grocery stores. Our motivation arises from the urgent need of store owners. In real world scenarios, even though many grocery items have been tagged with price, the semi-intervention of store employee is still essential, especially for vegetables and fruits, due to their various packaging and pricing methods. Another factor that delays the process is the strict requirement for grocery item label to align with the electronic scanner during checking out. Most of the customers and newly-hired service employees find it hard to align the price label with electronic scanner. By using the automatic recognition of grocery items, most of the tagging and scanning work can be saved, thus improving the overall efficiency of checking out and reduce human labor cost.

According to the survey [110], the in-store technologies are changing over the years, including evolving transformations, such as hand held scanners, the use of smartphone, artificial intelligence and geofencing technology heralded by Amazon Go [111]. In modern automated retail store like Amazon Go, customers are able to purchase products without being checked out by a cashier or using a self-checkout station [112]. The just-walk-out technology behind such facility involves computer vision, deep learning algorithms and sensor fusion. Under such advanced technology, the system depends heavily on the deployment of multiple sensors, especially various types of cameras. By using images and videos captured by the camera, the deep learning algorithm can accurately predict the product types and quantities. Among all such technologies, convolutional neural networks (CNN) have become the state-of-art method in various computer vision tasks, especially in the field of image classification. Various CNN architectures [10, 21, 12] have been proposed and shown outstanding performance in public datasets, including ImageNet [65], Microsoft Coco [73] and OpenImage [113]. The success of such CNN-based algorithm and applications relies on the rapid advancement in computing hardware and well-annotated datasets.

Existing datasets mainly consist of general objects and few of them focus on vegetables and fruits. As the urgent need for automatic classification of vegetable and fruit in retail industry increases, researchers have made great efforts to build large-scale, well-annotated fruit datasets. Vegfru [114] is one of the most recent large-scale domain-specific datasets for fruits and vegetables. However, algorithm design and application are still lagging behind the development of dataset. In this chapter, we adopt the edge computing paradigm to realize fast and accurate on-device image classification on mobile platforms. Our system is composed of two parts: 1) training deep learning models on the server using a compressed neural network; 2) developing and deploying an application running on mobile devices for fast and accurate inference. We develop an Android app to demonstrate the on-device CNN-based image classification on mobile devices, as depicted in Figure 6.1. To the best of our knowledge, this is the first attempt to deploy a large-scale CNN network for fruit and vegetable recognition on mobile devices. The main contributions of our research include:

1. We study and evaluate different CNN architectures and training parameters to get higher accuracy compared with other methods;

2. We use model compression and quantization to optimize the model for mobile

and embedded devices;

3. We design and develop an Android app for model inference and image classification, and evaluate the response time, memory consumption and CPU usage on the device. Our extensive experiment results show that our system can achieve excellent recognition accuracy with limited computation resources in a short time frame. Our application has shown great potential in the retail industry and supermarkets.

4. We conduct application study in real world scenarios in the grocery store and supermarkets in China and USA by collecting the customized fruit and vegetable image dataset in these areas separately and retrain our models using proposed approach. Our new model shows outstanding result in these application and provides potential to be deployed in real world retail stores and supermarkets.

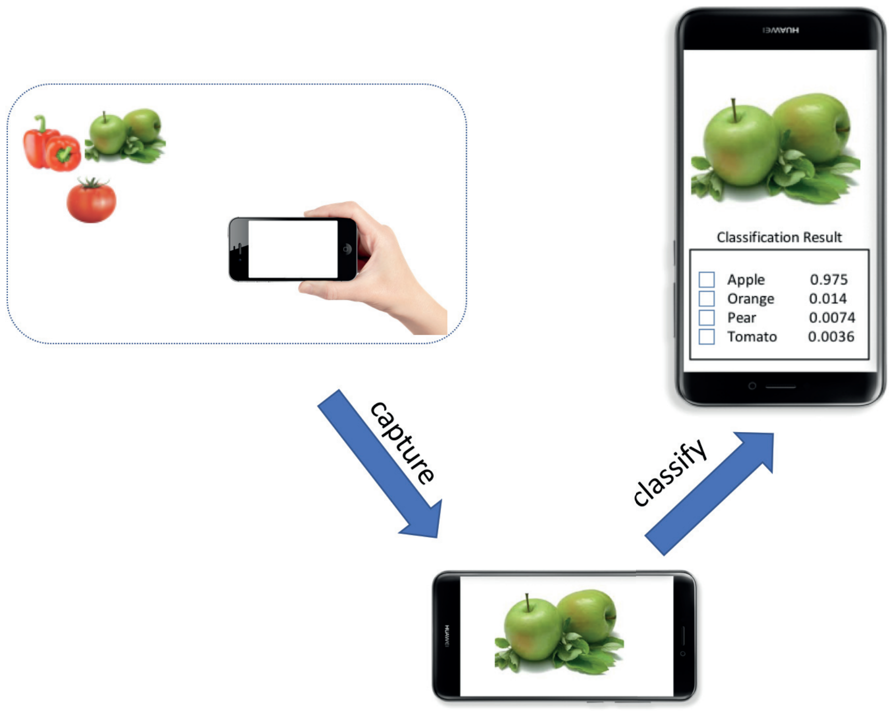


Figure 6.1: System demo for classifying vegetable and fruit using mobile device.

6.2 Background and Related work

There has been rising interest in using CNN-based model to improve the accuracy and reduce the response time for mobile vision applications. Existing work on mobile vision can be divided into three categories: model-based vision application [115], device-based mobile system [29] and edge-based computing paradigms [116].

6.2.1 Model-based Application

Recent years have witnessed the explosive advances in deep learning models. With the development of hardware, it's now possible to run CNN models on the mobile devices and lightweight embedded devices. The efforts in this field can be divided into two categories. The first category is to devise novel network architecture that exploit computation and memory efficient operation. Howard et al. [117, 115] proposed MobileNet that utilize the depth-wise and point-wise convolution to build lightweight CNNs and reduce inference time. ShuffleNet [118] used point-wise group convolution and channel shuffle, to greatly reduce computation cost while maintaining accuracy. DenseNet [119] proposed another structure that connects each layer to every other layer in a feed-forward fashion which substantially reduce the number of parameters with high accuracy. The second category is to use compression and quantization techniques to compress CNN models to reduce its resource demands. Jacob et al. [120] proposed a quantization scheme that allows inference to be carried out using integer-only arithmetic, which can be implemented more efficiently than floating point inference on integer-only hardware. Raghuraman et al. [121] presented techniques of quantizing the convolutional neural networks for inference with integer weights and activations. DeepCompression [122] also proposed a three stage processing pipeline: pruning, trained quantization and Huffman coding to reduce the storage and memory constrains.

6.2.2 Device-based Mobile System

There has been a significant amount of research going on and plenty of them focus on studying the system performance of deep learning system running on the mobile devices. Previous work on system research focus on the overall evaluation metrics when conducting the recognition tasks, which include accuracy, inference time, response time and power consumption [24, 61]. The studied system can be divided into two categories. The first category focus on studying the combination of cloud-based server and mobile devices [25, 26]. Researchers developed the system using mobile and cloud server together. Chen et al. [25] proposed to use mobile-edge and cloud services for system integration and studied the influence of computation offloading for multiple-users. Kang et al. [27] designed a lightweight scheduler to automatically partition DNN computation between mobile devices and data centers at the granularity of neural network layers to reduce latency and power consumption. The second category focus on developing system without cloud intervention. Keiji and Austin et al. developed a system [28, 29] for food recognition using CNN architecture and running the inference on-device separately. Latifi et al. [30] designed a system called CNNDroid for running CNN models on Android devices with GPU-accelerated execution. The device-based mobile approaches studied the system performance in real-world scenarios and provide complete evaluation and guideline for deployment.

6.2.3 Edge-based Computing Paradigms

Edge computing [31] usually refers to the enabling technology that allows computation to be performed at the edge of the network. The "edge" devices can be any computing and network resources along the path between data sources and cloud data center. The data source can be any sensing devices like smartphone, smartwatch, PDA and tablet that collect sensor data like image, audio and video. Cloud data center is equipped with powerful servers that can perform complex computation and data processing. By utilizing the computation ability of edge devices, we can address the critical issues of response time requirement, battery life constraint, bandwidth cost saving, as well as data safety and privacy. The major challenge [26, 123] in applying deep learning algorithms and building visual categorization system is to devise a high-performance mechanism that utilizes the edge computing power for accurate recognition within limited response time and computation resources.

Most of the current research for automatic categorization of vegetable and fruit images are focused on two aspects. The first aspect is to build a large scale, well-annotated high-quality dataset [124, 125, 126], providing the foundation for designing well-structured deep learning models and neural networks. The second aspect is to design a highly efficient algorithm that utilize the previous dataset and achieve the state-of-art precision or speed. Recent years have witnessed an increase of available datasets. Hou et al. [114] have collected and annotated the largest fine-grained vegetable and fruit dataset VegFru that contains 292 categories. Other efforts in developing such dataset include the Fruits-360 [127], DeepFruit [128], Food-101 [72], UEC-256 [129], but they only contain a small number of categories and images. With the development of well-annotated dataset, the next big challenge for developing accurate algorithms for categorization is to address the inter-class and intra-class differences within the dataset. Recent progress in convolutional neural network [10, 21, 12, 66] has shown significantly better performance than hand-engineered features (shape, color and texture) [124, 130].

In this chapter, we adopt the structure and computing paradigm of edge computing and develop our edge-computing based system based on convolutional neural network. Our efforts are divided into three parts. We first train our customized neural network based on MobileNet [117, 115] which could reduce the computation and inference time. Then we compress and quantize the trained model using tensorflow for mobile deployment. Furthermore, we implement the model inference and develop an Android application to classify the image and video in real time.

6.3 Proposed Approach

The ultimate goal of our research is to design and deploy a reliable and efficient system to classify various vegetable and fruit images within limited time and computation capability constraints. The main contributions of our research include: 1) We study and evaluate different CNN architectures and training parameters to get higher accuracy compared with other methods; 2) We use model compression and quantization to optimize the model for mobile and embedded devices; 3) We design and develop an Android app for model inference and image classification, and evaluate the response time, memory consumption and CPU usage on the device.

6.3.1 Convolutional Neural Network

Convolutional Neural Network is widely used in computer vision tasks, such as image classification, object detection, and visual question answering. A convolutional neural network is usually composed of convolutional layers, pooling layers, and fully-connected layers. Each layer is connected to the previous layer by predefined, fix-sized kernels. The weights or parameters within each layer are shared to reduce computation complexity. By using carefully designed network architecture, the CNN model learns the parameters from a large-scale dataset to represent the global and local features in the images without using hand-crafted features. Every model has various types of layers and activation function as well as different number of layers and connections that can exhibit strong hidden feature representation ability than human-engineered features. More details of the network structure can be found in [10, 57, 1].

Most of previous CNN models are designed for desktop computer or servers with reasonable CPU, memory and GPU support. In this chapter, we explore various CNN models suitable for mobile devices. While general CNN models such as AlexNet [10], GoogLeNet [12], ResNet [66] show good results for the large benchmark dataset ImageNet, for domain-specific fine-grained dataset VegFru, due to the minor difference between different classes, the models fail to achieve the best performance. Another drawback of such general models is that they contain a large number of parameters, making it unsuitable to be deployed directly on the mobile devices due to the long inference time and large memory footprint. To tackle such problems, we dive deep into the neural network structure and propose several training strategies to achieve the best result with limited resources.

MobileNet. MobileNet is a light-weight convolutional neural network that uses depth-wise separable convolutions to build network specifically for mobile and embedded devices. Normally it contains 6 convolutional blocks $conv_1$, $conv_2$, ..., $conv_6$, as shown in Fig.6.2. Each block contains some 3x3 kernels with a stride size of 1 or 2. $conv_1$ is a convolutional layer with 3x3 kernel and the stride size is 1. For $conv_2$, $conv_3$, $conv_4$ and $conv_6$, they contain repeated units such as two depthwise (DW) convolution layer with a 3x3 kernel and the stride size is 1 and 2 separately. $conv_5$ contains 5 layers of depthwise convolution. The whole network is formed by appending a fully-connected layer and softmax layer to the convolution structure, producing a classification score for each pre-defined class in the dataset. The last softmax layer has a fixed size that equals to the class number. Here we use 292 for the whole Vegfru dataset. A depth-wise separable convolution is a form of factorized convolutions that factorize a standard convolution into a depth-wise convolution and a 1x1 convolution. As illustrated in Fig.6.2, each unit in the dotted rectangle has similar structure as the right module, which contains a 3x3 depthwise convolution layer and 1x1 pointwise convolutional layer, with one batch normalization (BN) layer and rectified linear unit (ReLU) appended after each convolutional layer in thie module. By using this

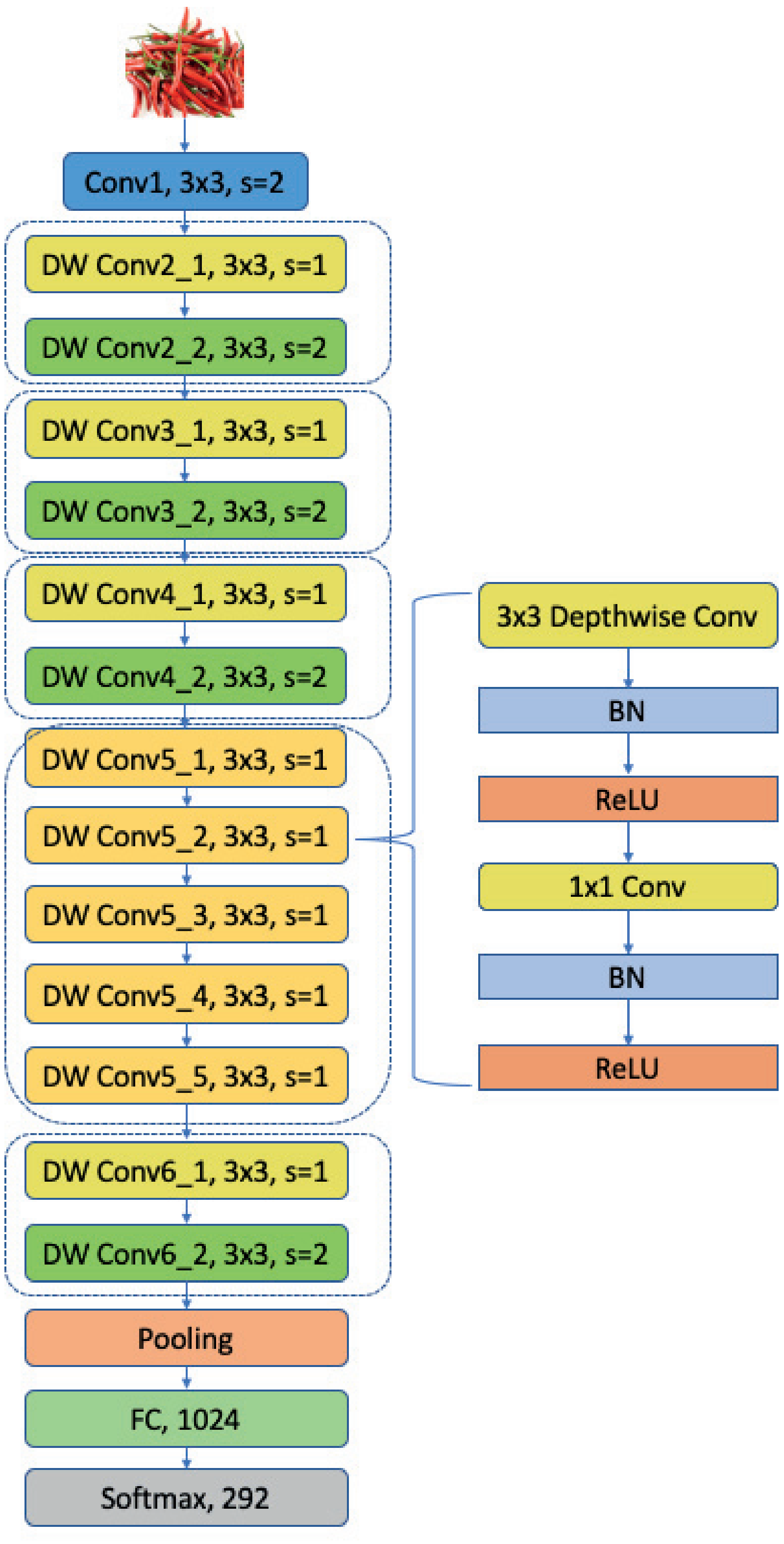


Figure 6.2: Proposed method using MobileNet.

convolution to replace standard convolution, a new network structure is formed as MobileNet.

6.3.2 Transfer Learning

Transfer learning [131, 102] is a popular method in computer vision that helps to build accurate models in a time-saving way. As shown in Fig.6.3, transfer learning aims to apply the learned knowledge from one domain to another different but related domain. Instead of starting the learning process from scratch, it starts the learning process from previous pretrained model, which usually requires large number of images to learn powerful features, thus reducing the learning time and requirement for large-scale domain-specific dataset. A pretrained CNN model on ImageNet has been widely used for transfer learning, either by using pretrained network as a feature extractor or using it to finetune the whole network. During finetuning, the model can still learn powerful weights to represent image features without too much training data. The reason behind such a training strategy is that the CNN model gains general representation ability from pretrained model on natural images. After finetuning, the model adjusts the parameters to represent the unique features in the target dataset. Experimental results show that transfer learning is very effective on major public datasets [132]. When finetuning a model, there are usually several kinds of strategies to achieve the best result. The first strategy is to train the entire model that updates all the parameters in every layer. This usually requires a large number of training images. Another strategy is to train some layers and leave the others unchanged, or freeze the whole convolutional base and only update the fully-connected layers. This latter strategy is suitable for large dataset with small amount of model parameters. In our experiment, we replace the last softmax classifier layer with our predefined layer and then explore all possible strategies. Our model is pretrained from ImageNet dataset and finetuned on our Vegfru dataset to get the best accuracy using those training strategies to achieve the best result.

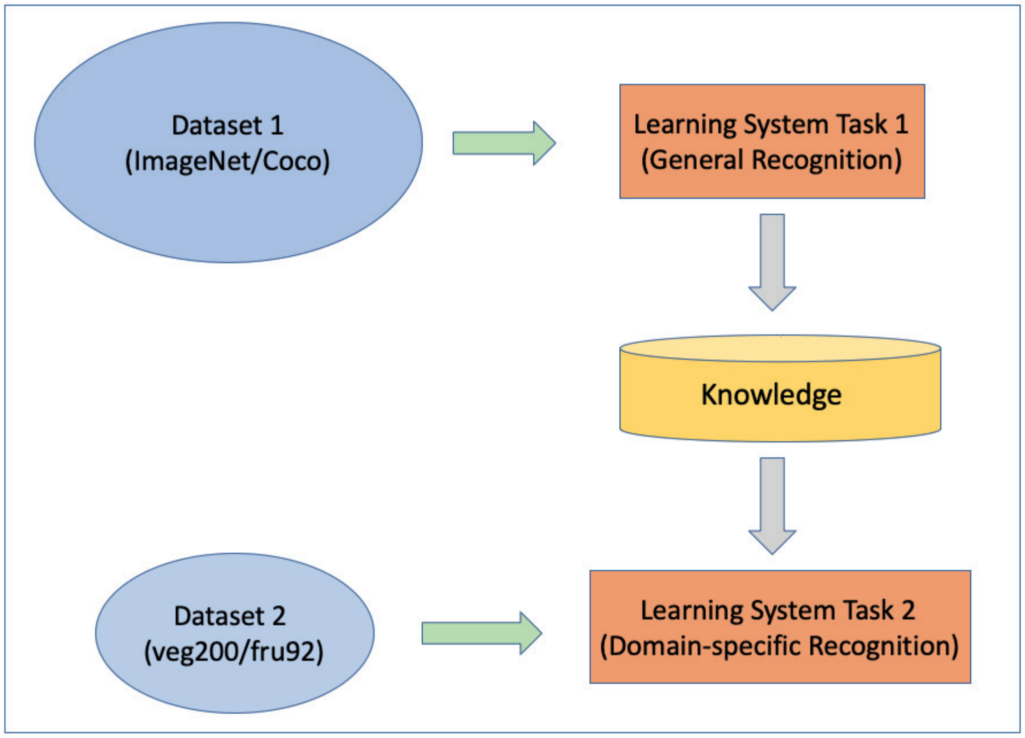


Figure 6.3: Transfer learning - training on small domain-specific dataset from pre-trained model.

6.3.3 Model Quantization

Mobile quantization [122, 121] refers to the technique that allows for the reduced precision representation of weights and activations for both storage and computation. As a result of quantization, memory access for reading and storing intermediate activations are largely reduced. As shown in Fig.6.4.

To provide practical support for general models, we use Tensorflow [133] to implement our algorithm and neural network. Furthermore, we conduct post-training quantization that quantizes that weights and activations. The post-training quantization quantizes weights to 8-bits of precision from floating point, thus reducing the model size and providing up to 3x speed up with little degradation in model accuracy. As shown in Fig.6.4, the dotted rectangle depicts the added module for weights

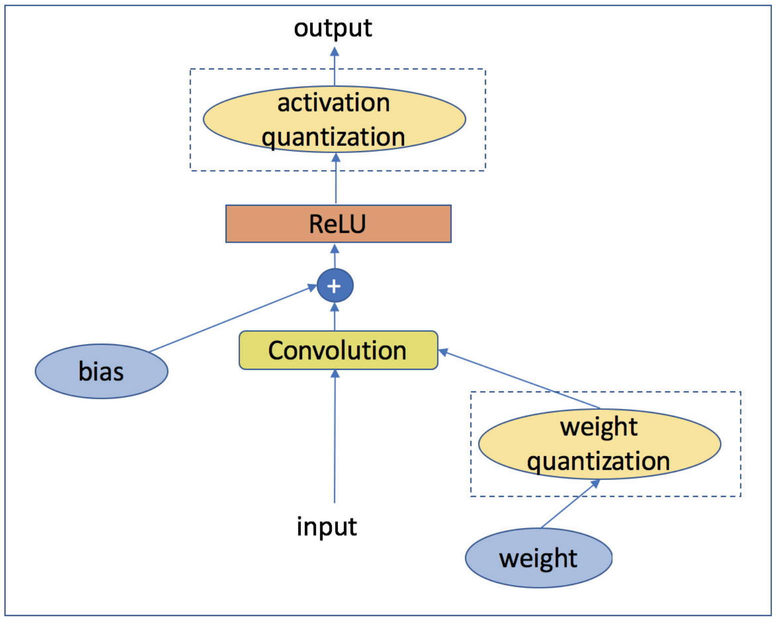


Figure 6.4: Model quantization.

and activation quantization. After the quantization, the computation and in-memory storage are significantly reduced. Here we use Tensorflow-lite (TF-Lite) to conduct such a conversion. More specifically, TF-Lite provides built-in models and quantization tools called TOCO to convert trained tensorflow model to a compressed format **.tflite*. Such quantization and on-device inference will use the 8-bit instead of floating point operation on the original model. Our experiment result shows that such a conversion improves the speed significantly.

6.4 Experiments

We conduct extensive experiments on the public Vegfru image dataset and follow the standard evaluation process. The model accuracy, memory consumption and response time are compared for several different models.

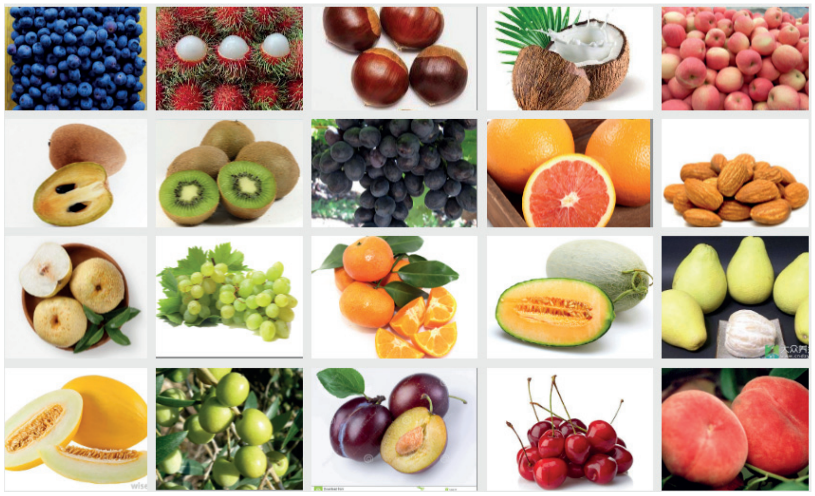


Figure 6.5: Fru92 dataset images.

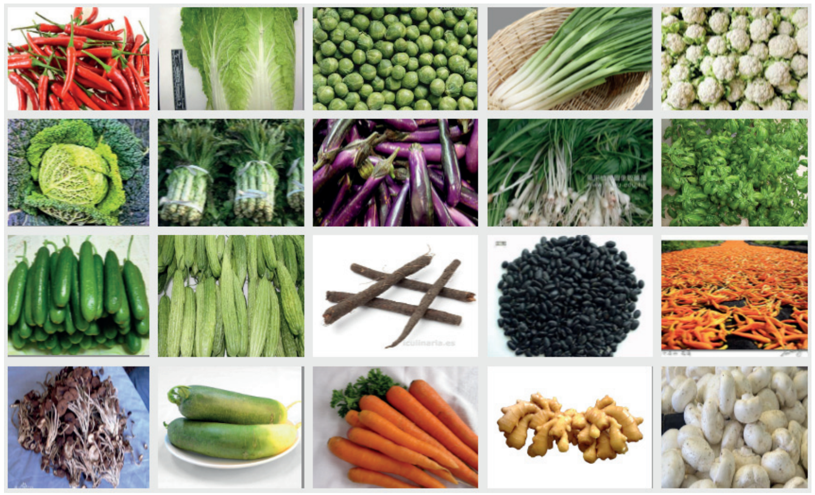


Figure 6.6: Veg200 dataset images.

6.4.1 Dataset Details

Due to the limitation of publicly available vegetable and fruit datasets, we only conduct our experiment on Vegfru [114] dataset, as it is the first large-scale, well-annotated dataset. This dataset covers vegetables and fruits with 25 upper-level categories and 292 sub-level categories, accounting for more than 160,000 images in total. For vegetables, it has 15 sup-classes with 200 sub-classes and every category has a maximum of 1,807 images and a minimum of 202 images. For fruits, it contains 10 sup-classes and 92 sub-classes and every category has a maximum of 1,615 images and a minimum of 202 images. There are 91,117 images for vegetables and 69,614 images for fruits in total. Fig. 6.5 and Fig. 6.6 show the thumbnails for images from the fruit and vegetable category separately. Table 6.1 illustrates the characteristics of the dataset, with the number of images for each sub-class varying from 200 to 2,000. Here we also name the vegetables as Veg200 and Fru92 for the fruits. In the following section, we will evaluate the accuracy of our model on these datasets separately.

	#Sup	**#Sub**	**#Min**	**#Max**
Vegetables	15	200	202	1807
Fruits	10	92	202	1615

Table 6.1: Data distribution in VegFru dataset.

6.4.2 Hyperparameter Tuning

We finetune our model based on the pretrained MobileNet model from ImageNet. For the last layer of the MobileNet, we place a softmax layer to get the confidence score for each category. Each score ranges from 0 to 1, representing the probability of classifying each class correctly. When training our MobileNet model, dropout and ReLU are used to address the overfitting and convergence issues. We use Tensorflow and TF-Slim to implement our neural network architecture. The model is trained using stochastic gradient descent (SGD) with different combinations of parameters.

For MobileNet, we choose two hyper-parameters, width multiplier and resolution multiplier. Width multiplier α is defined to thin a network uniformly at each layer. α ranges from 0 to 1 with a typical value of 0.25, 0.5, 0.75 and 1. $\alpha = 1$ is the baseline MobileNet model and smaller α defines thinner models. The resolution multiplier ρ is used to reduce image size. In practice, we usually set the input size implicitly to be 128, 160, 192, 224 for different ρ values. By experiments, we find that using larger image resolution will contribute to the increase of accuracy without introducing too much latency. We use 224x224 as the input image size for training the neural network.

6.4.3 Implementation

Our experiment contains two parts: the server side and client side. In the first stage, we train our models in various settings with different parameters and choose the best performer as the final selected model. We train all the models on multiple NVIDIA Tesla K80 GPUs using Tensorflow. During training, the input image is randomly cropped from the original image and resized to a fixed input size with scale. We train all networks using the RMSProp optimizer with a momentum of 0.9, and the batch size of 32. The initial learning rate is 0.045, with exponential decay of 0.98 per epoch. We use batch normalization after every layer, and the standard weight decay is set to be 0.00004.

In the second stage, we convert and quantize the trained-well tensorflow model to TF-Lite format to reduce the model size and shorten inference time. Then we build an Android application that follows the standard testing procedure. An image is first preprocessed before being fed into the neural network. The original image is center cropped and resized to the target input size 224x224. Then we remove the mean value for each pixel afterward. The model inference is implemented in Java on the Android devices using *libtensorflow* and other built-in Android libraries for image preprocessing.

6.4.4 Model Evaluation

We classify all the classes for the whole datasets (VegFru, Veg200, Fru92). For each dataset, we evaluate the performance on the super-classes and their sub-classes. To make a fair comparison, we strictly follow the same data splits as in VegFru. For each subclass, the first 100 images are selected as train set, the following 50 images as val set, and the rest as test set. As released in the original dataset, the image lists are used to construct the tfrecords for training our models. All models are evaluated with the *test* set. The results are shown in Table 6.2. Note that due to the missing splits and image lists (Veg200's super-class and Fru92's super-class), the result on such two sets is not available in our experiment.

Table 6.2 shows that our proposed MobileNet model can achieve the state-of-art accuracy in all datasets. The classification accuracy on super-class is usually better than on sub-class due to the more significant difference in super-class and the minor intra-class difference in sub-class. This single model is trained in one-stage without any model ensemble, which makes the inference faster and more suitable for mobile devices.

Dataset	**Category**	**CaffeNet**	**VGGNet**	**GoogLeNet**	**MobileNet**
Veg200	15 sup-classes	74.92%	83.81%	83.50%	-
	200 sub-classes	67.21%	78.5%	80.17%	**82.26%**
Fru92	10 sup-classes	79.86%	86.81%	87.54%	-
	92 sub-classes	71.60%	79.80%	81.79%	**83.43%**
VegFru292	25 sup-classes	72.87%	82.45%	82.52%	**82.72%**
	292 sub-classes	66.40%	77.12%	79.22%	**81.72%**

Table 6.2: **Baselines on VegFru for CNN models.** The typical CaffeNet, VGGNet, GoogLeNet are chosen as baselines while MobileNet are compared with the previous state-of-art results.

Table 6.3 shows that our proposed MobileNet model can achieve the state-of-art accuracy in all datasets over other feature-based machine leanring methods. Here we use PHOW features as our feature descriptors. As shown in [106], PHOW features are a variant of dense SIFT descriptors extracted from multiple scales. Using the same splits of training and testing dataset as for the CNN-based method, we first build the feature vocabulary, then compute the spatial histograms and generate the bins for feature encoding, and finally generate the feature maps. In the last step, a SVM and random forest classifier are trained based on the feature maps. For

SVM implementation, we use VLfeat's MATLAB imeplementation[1]. SGD and SGCA solver are chosen separately to get the best result. For random forest, we try different configurations of parameters and find that a maxium number of 100 tress and a max depth of 9 generate the best result. Experimental results show that our proposed CNN model achieves best classification accuracy and outperforms these traditional feature based classifiers by a large margin.

	Veg200	**Fru92**	**Vegfru292**
PHOW+SVM (sgd, [106])	24.54%	26.41%	18.45%
PHOW+SVM (sgca, [106])	28.81%	30.33%	26.06%
PHOW+RF	40.73%	43.21%	38.01%
MobileNet(proposed)	**82.26%**	**83.43%**	**81.72%**

Table 6.3: **Baseline on Vegfru for other classifiers.** The typical SVM, Random Forest (RF) are chosen as baselines while MobileNet are compared with the previous state-of-art results.

6.4.5 System Evaluation

In the second stage, we evaluate our system on mobile devices. As shown in Table 6.4, our model is first converted to a frozen graph file to store the graph definition of the neural network structure. We remove the nodes that are not called during inference, shrink expressions that are constant into single nodes and optimize away some multiplication operations during batch normalization by pre-multiplying the weights for convolutions. Then we use *TF-Converter* to convert the frozen graph to TF-Lite format for reducing storage and inference. The result shows that the model size decreases by more than 60% from the original tensorflow model. We also evaluate our MobileNet and Inception-v3 model derived from the GoogLeNet on a UnisCom MZ96-Plus tablet equipped with Intel Z23735F Quadro cores. As shown in Table 6.5, the inference time, storage and memory consumption are significantly reduced compared with other baseline models without sacrificing the classification

[1]http://www.vlfeat.org/overview/svm.html#tut.svm

accuracy. For simple baseline models like AlexNet and VGGNet, our model can even outperform them with much less computation resources.

	Checkpoint	**Frozen graph**	**TF lite**
Veg200	29MB	9.8MB	9.5MB
Fru92	27MB	9.2MB	8.9MB
Vegfru292	31MB	11MB	9.9MB

Table 6.4: **Model size on Vegfru.** The trained model is converted to frozen graph and quantized to TF-Lite format.

Model	**Time**	**Storage**	**Memory**	**CPU**
Inception-v3(lit)	3291ms	84MB	129MB	41%
Inception-v3(lit-quant)	2890ms	22MB	109MB	33%
MobileNet(lit)	573ms	11MB	74MB	31%
MobileNet(lit-quant)	462ms	9.9MB	67MB	27%

Table 6.5: System evaluation on MobileNet and baseline model for time, storage, memory consumption and CPU utilization on mobile devices.

The proposed approach and system has three characteristics: First, the system can provide potential deployment solution for building automatic vegetable and fruit recognition solution in retail store. Second, the algorithm and method presented here utilize memory and computation efficient CNN models for mobile devices that can be applicable to other embedded devices and intelligent systems. Third, the system provides stable service, fast response and high accuracy.

6.4.6 Application Study

To verify the effectiveness of our proposed system, we collaborate with supermarkets for deployment in real application scenarios. Even though our model improves the accuracy significantly, there is some gap between machine-based vision recognition and human recognition. To overcome such gap, we dive deep into the specific problem and dataset. There are several factors that causes the deterioration of accuracy: 1) the minor difference within the same super class. For example, in the original VegFru292 dataset, there are multiple kinds of mushrooms that belong to different categories. However, in real world scenarios, there are only limited kinds of mushrooms available in specific area and supermarkets; 2) the random noises in the image dataset, especially for the misclassified label, text occlusion and other objects in the foreground and background; 3) the various angle, environment and exposure in the images crawled from the internet.

To address the aforementioned issues, we explore two kinds of efforts. Firstly, we ask our collaborator in China to collect a new dataset with their smartphone and tablet. Such images are taken in real grocery stores and only contain limited kinds of categories. We use such dataset to train and evaluate our model's accuracy. Such efforts will help to reduce noise and reduce the number of categories, thus improving the overall accuracy, as shown in Table 6.6. Our dataset is named GroCN20 which contains 20 categories of very common vegetables in grocery stores in China. There are 13,192 images in total, with 80% of them used for training and the rest used as the test set. Our experiment result shows that our model can achieve a very high accuracy with nearly no false classification. Such model and dataset development will help to boost the training accuracy and speed up the deployment in real world application settings.

Second, to study the feasibility in the United States, we visit various grocery stores in the US and choose the corresponding vegetable and fruit categories. Since VegFru292 contains many categories that only appear in specific countries and areas, we manually choose 26 categories of fruits and 46 categories of vegetables that appear in the supermarket Market Basket [2] from the original VegFru dataset. Most of the fruits and vegetables in Market Basket are covered in our new dataset. We name these two datasets as FruMB26 and VegMB46 separately and combine them as VegFruMB72. We use the same approach and strategy to train and evaluate our model.

[2]http://www.mygrocerychecklist.com

The new dataset still use the 80/20 splits as the training and testing set. As shown in Table 6.6, our model shows excellent accuracy boost on this new dataset and provides promising path for future deployment in real grocery store and supermarkets.

	# TrainSet	**# TestSet**	**# Total**	**Accuracy**
GroCN20	10,554	2,638	13,192	**99.30%**
FruMB26	16,424	4,106	20,530	**97.05%**
VegMB46	24,586	6,146	30,732	**96.30%**
VegFruMB72	41,010	10,252	51,262	**96.52%**

Table 6.6: Evaluation on customized grocery dataset using proposed approach.

6.5 Conclusion

We designed and implemented an edge computing visual system to classify vegetables and fruits on the domain-specific VegFru dataset. Our work is the first research attempt using CNN in mobile devices for a large vegetable dataset. Based on the experiment result, our system has shown outstanding performance in image categorization with limited memory consumption in a short time frame. The next step is to collaborate with engineering researchers to deploy such a system in retail stores and measure its performance in realistic scenarios. We will also use more labeled region-based information for accurate localization to further improve accuracy and handle extreme cases in real-world settings. Our system has provided a promising path for automatic item categorization in self-service supermarkets.

Chapter 7

CONCLUSION

In this thesis, we present the deep-learning based image recognition system on an edge computing service infrastructure. We study how to combine deep learning algorithm and edge computing devices for accurate image recognition under the strict constraint of latency, accuracy and battery consumption. Our proposed approach and system show outstanding performance in various system metrics in different application scenarios, especially in the healthcare domain for dietary assessment, medical image analysis and tuberculosis diagnosis. We propose different deployment schemes for these applications.

We first research on how to conduct dietary assessment on edge devices, where the deep learning model and image analysis algorithms are deployed on a cloud server and mobile devices separately. To do this, we first investigate on the blurry detection and image segmentation using traditional feature based algorithms. After filtering the blurry image and get the segmented patches from original clear image, we send them into the cloud server. These pipelines are implemented on Android using Java and OpenCV. In the cloud server, we train our image classification model and fine-tune it from pretrained model on ImageNet dataset. We design our convolutional neural network and implement the AlexNet and GoogLeNet model using Caffe. Our model is deployed on the server with a web server implemented by Django and Python, thus listening to a HTTP port for receiving the image classification request. After receiving the image, we run the model inference and predict the label for the input image. The final result is sent back to the edge device and displayed on the screen. The system's response time, power consumption and system accuracy are evaluated during this whole process.

Next, we explore how deep learning and mobile techniques can be leveraged to improve the tuberculosis diagnosis in resource poor and marginalized area. We propose to use deep learning algorithms to improve the diagnosis accuracy and deploy the system on mobile devices to speed up the diagnosis and reduce the patients' wait time. To implement such ideas, we develop a set of algorithms and system to assist data collection and model training. First, we collaborate with a team formed by computer scientist, physicians and doctors to collect chest X-ray images from the hospital. Then we build an annotation software that can help us collect the label and bounding box information to further facilitate the training of various deep learning models. At last, we preprocess all the chest X-ray image from the dataset and train several deep learning based models. Such models are deployed on the mobile devices and cloud server to help field study in Perú. Our evaluation and discussion with the clinical experts have show that the proposed method using deep learning and mobile techniques could help the tuberculosis diagnosis and speed up the whole process.

We also investigate how the combination of deep learning and edge computing could leverage the automatic checkout process in retail industry, especially in the supermarkets and grocery stores. Specifically, we research how to deploy an edge computing system in grocery store to eliminate the wait time and facilitate the checkout process. First, we collect vegetable and fruit images from online dataset and actual grocery environment in the US and China. Then, we design and build convolutional neural network, specifically for mobile and embedded devices. Our CNN model adopts the architecture of MobileNet and shows outstanding performance in CPU, memory and storage utilization without sacrificing too much accuracy and processing time. We train our deep learning model and quantize the model for mobile devices running on Android system. The system performance are evaluated in various metrics to study the application potential in different environmental settings. Additionally, we collaborate with Chinese supermarkets and collect another dataset for most commonly vegetable and fruits there. We also visit many US supermarkets and collect their vegetable and fruit categories to select enough image data from a commonly used large-scale dataset named VegFru. Our system and model show significant improvement over other hand-engineered feature based machine learning classifiers and CNN models on these datasets within limited computation and computational resources.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Khotanzad and Y. H. Hong, “Invariant image recognition by zernike moments,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 5, pp. 489–497, 1990.

[3] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *international Conference on computer vision & Pattern Recognition (CVPR’05)*, vol. 1. IEEE Computer Society, 2005, pp. 886–893.

[4] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[6] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[7] S. Leutenegger, M. Chli, and R. Siegwart, “Brisk: Binary robust invariant scalable keypoints,” in *2011 IEEE international conference on computer vision (ICCV)*. Ieee, 2011, pp. 2548–2555.

[8] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[9] A. Alahi, R. Ortiz, and P. Vandergheynst, “Freak: Fast retina keypoint,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2012, pp. 510–517.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[12] C. Szegedy, W. Liu, Y. Jia, Sermanet *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[13] L. Wang and D.-C. He, “Texture classification using texture spectrum,” *Pattern Recognition*, vol. 23, no. 8, pp. 905–910, 1990.

[14] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[15] I. Daubechies, *Ten lectures on wavelets*. SIAM, 1992.

[16] F. Schaffalitzky and A. Zisserman, “Viewpoint invariant texture matching and wide baseline stereo,” in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2. IEEE, 2001, pp. 636–643.

[17] U. Avni, H. Greenspan, E. Konen, M. Sharon, and J. Goldberger, “X-ray categorization and retrieval on the organ and pathology level, using patch-based visual words,” *IEEE Transactions on Medical Imaging*, vol. 30, no. 3, pp. 733–746, 2011.

[18] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and Y. Ma, “Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment,” in *International Conference on Smart Homes and Health Telematics*. Springer, 2016, pp. 37–48.

[19] Y. Bengio, L. Yao, G. Alain, and P. Vincent, “Generalized denoising autoencoders as generative models,” in *Advances in neural information processing systems*, 2013, pp. 899–907.

[20] R. Salakhutdinov and G. Hinton, “Deep boltzmann machines,” in *Artificial intelligence and statistics*, 2009, pp. 448–455.

[21] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[22] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[23] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[24] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing?" in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications.* ACM, 2015, pp. 117–122.

[25] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[26] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 249–261, 2018.

[27] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1. ACM, 2017, pp. 615–629.

[28] K. Yanai, R. Tanno, and K. Okamoto, "Efficient mobile implementation of a cnn-based object recognition system," in *Proceedings of the 24th ACM international conference on Multimedia.* ACM, 2016, pp. 362–366.

[29] A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. P. Murphy, "Im2calories: towards an automated mobile vision food diary," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1233–1241.

[30] S. S. Latifi Oskouei, H. Golestani, M. Hashemi, and S. Ghiasi, "Cnndroid: Gpu-accelerated execution of trained deep convolutional neural networks on android," in *Proceedings of the 2016 ACM on Multimedia Conference.* ACM, 2016, pp. 1201–1205.

[31] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[32] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Optical Pattern Recognition XII*, vol. 4387. International Society for Optics and Photonics, 2001, pp. 95–102.

[33] P. E. Lestrel, *Fourier descriptors and their applications in biology.* Cambridge University Press, 1997.

[34] T. Tan, "Rotation invariant texture features and their use in automatic script identification," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 7, pp. 751–756, 1998.

[35] O. D. Faugeras and M. Hebert, "The representation, recognition, and locating of 3-d objects," *The international journal of robotics research*, vol. 5, no. 3, pp. 27–52, 1986.

[36] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[39] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: at the leading edge of mobile-cloud convergence," in *6th International Conference on Mobile Computing, Applications and Services*. IEEE, 2014, pp. 1–9.

[40] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[41] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, and G. Tamm, "Smart items, fog and cloud computing as enablers of servitization in healthcare," *Sensors & Transducers*, vol. 185, no. 2, p. 121, 2015.

[42] Y. Cao, C. Liu, B. Liu, M. J. Brunette, N. Zhang, T. Sun, P. Zhang, J. Peinado, E. S. Garavito, L. L. Garcia *et al.*, "Improving tuberculosis diagnostics using deep learning and mobile health technologies among resource-poor and marginalized communities," in *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE, 2016, pp. 274–281.

[43] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, 2009.

[44] Y. Cao, S. Chen, P. Hou, and D. Brown, "Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation," in *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*. IEEE, 2015, pp. 2–11.

[45] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu *et al.*, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[46] P. Maduskar, L. Hogeweg, R. Philipsen *et al.*, "Improved texture analysis for automatic detection of tuberculosis (tb) on chest radiographs with bone suppression images," in *SPIE Medical Imaging*. International Society for Optics and Photonics, 2013.

[47] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention.* Springer, 2015, pp. 234–241.

[48] C. L. Ogden, M. D. Carroll, B. K. Kit, and K. M. Flegal, “Prevalence of childhood and adult obesity in the united states, 2011-2012,” *Jama*, vol. 311, no. 8, pp. 806–814, 2014.

[49] W. H. Organization, W. H. Organization *et al.*, “Obesity and overweight fact sheet. 2016,” *URL: http://www. thehealthwell. info/node/82914*, 2017.

[50] G. H. Beaton, J. Milner, P. Corey, V. McGuire, M. Cousins, E. Stewart, M. De Ramos, D. Hewitt, P. Grambsch, N. Kassim *et al.*, “Sources of variance in 24-hour dietary recall data: implications for nutrition study design and interpretation,” *The American journal of clinical nutrition*, vol. 32, no. 12, pp. 2546–2559, 1979.

[51] J. Cade, R. Thompson, V. Burley, and D. Warm, “Development, validation and utilisation of food-frequency questionnaires–a review,” *Public health nutrition*, vol. 5, no. 4, pp. 567–587, 2002.

[52] R. Steele, “An overview of the state of the art of automated capture of dietary intake information,” *Critical reviews in food science and nutrition*, vol. 55, no. 13, pp. 1929–1938, 2015.

[53] Y. Matsuda and K. Yanai, “Multiple-food recognition considering co-occurrence employing manifold ranking,” in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012).* IEEE, 2012, pp. 2017–2020.

[54] S. Yang, M. Chen, D. Pomerleau, and R. Sukthankar, “Food recognition using statistics of pairwise local features,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE, 2010, pp. 2249–2256.

[55] F. Zhu, M. Bosch, I. Woo, S. Kim, C. J. Boushey, D. S. Ebert, and E. J. Delp, “The use of mobile devices in aiding dietary assessment and evaluation,” *IEEE journal of selected topics in signal processing*, vol. 4, no. 4, pp. 756–766, 2010.

[56] C. K. Martin, T. Nicklas, B. Gunturk, J. B. Correa, H. R. Allen, and C. Champagne, “Measuring food intake with digital photography,” *Journal of Human Nutrition and Dietetics*, vol. 27, pp. 72–81, 2014.

[57] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT Press, 2016.

[58] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[59] S. Yi, Z. Hao, Z. Qin, and Q. Li, “Fog computing: Platform and applications,” in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb).* IEEE, 2015, pp. 73–78.

[60] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*. ACM, 2015, pp. 37–42.

[61] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 249–261, 2017.

[62] R. R. Wing and S. Phelan, "Long-term weight loss maintenance–," *The American journal of clinical nutrition*, vol. 82, no. 1, pp. 222S–225S, 2005.

[63] B. L. Daugherty, T. E. Schap, R. Ettienne-Gittens, F. M. Zhu, M. Bosch, E. J. Delp, D. S. Ebert, D. A. Kerr, and C. J. Boushey, "Novel technologies for assessing dietary intake: evaluating the usability of a mobile telephone food record among adults and adolescents," *Journal of medical Internet research*, vol. 14, no. 2, 2012.

[64] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[65] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[67] Y. Jia, E. Shelhamer, J. Donahue *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[68] J. Oh, S. Hwang, Y. Cao, W. Tavanapong, D. Liu, J. Wong, and P. C. De Groen, "Measuring objective quality of colonoscopy," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 9, pp. 2190–2196, 2009.

[69] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[70] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5353–5360.

[71] Y. Kawano and K. Yanai, "Foodcam: A real-time food recognition system on a smartphone," *Multimedia Tools and Applications*, vol. 74, no. 14, pp. 5263–5287, 2015.

[72] L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101–mining discriminative components with random forests,” in *European Conference on Computer Vision*. Springer, 2014, pp. 446–461.

[73] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[74] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, “Quantized convolutional neural networks for mobile devices,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4820–4828.

[75] W. H. Organization *et al.*, “Global tuberculosis report 2016,” *World Health Organization*, 2016.

[76] B. Caputo, T. Tommasi, H. Muller, T. M. Deserno, and J. Kalpathy-Cramer, “Imageclef 2009 lung nodule detection and medical annotation task,” in *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, 2009, pp. 72–84.

[77] J. Shiraishi, S. Katsuragawa, J. Ikezoe, T. Matsumoto, T. Kobayashi *et al.*, “Development of a digital image database for chest radiographs with and without a lung nodule: receiver operating characteristic analysis of radiologists’ detection of pulmonary nodules,” *American Journal of Roentgenology*, vol. 174, no. 1, pp. 71–74, 2000.

[78] S. Jaeger, A. Karargyris, S. Antani, and G. Thoma, “Detecting tuberculosis in radiographs using combined lung masks,” in *Engineering in Medicine and Biology Society*. IEEE, 2012.

[79] T. Xu, I. Cheng, R. Long, and M. Mandal, “Novel coarse-to-fine dual scale technique for tuberculosis cavity detection in chest radiographs,” *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, p. 3, 2013.

[80] Y.-L. Song and Y. Yang, “Localization algorithm and implementation for focal of pulmonary tuberculosis chest image,” in *Machine Vision and Human-Machine Interface*. IEEE, 2010.

[81] S. Jaeger, A. Karargyris, S. Candemir, J. Siegelman, L. Folio, S. Antani, G. Thoma, and C. J. McDonald, “Automatic screening for tuberculosis in chest radiographs: a survey,” *Quantitative imaging in medicine and surgery*, vol. 3, no. 2, pp. 89–99, 2013.

[82] E. Ruiz, Á. Proaño, O. Ponce, and W. Curioso, “Mobile health for public health in peru: lessons learned,” *Revista peruana de medicina experimental y salud publica*, vol. 32, no. 2, pp. 364–372, 2015.

[83] V. Mani, S. Wang, F. Inci, G. De Libero, A. Singhal, and U. Demirci, "Emerging technologies for monitoring drug-resistant tuberculosis at the point-of-care," *Advanced drug delivery reviews*, vol. 78, pp. 105–117, 2014.

[84] M. Zimic, J. Coronel, R. H. Gilman, C. G. Luna, W. H. Curioso, and D. A. Moore, "Can the power of mobile phones be used to improve tuberculosis diagnosis in developing countries?" *Transactions of the Royal Society of Tropical Medicine and Hygiene*, vol. 103, no. 6, pp. 638–640, 2009.

[85] A. B. Schwartz, G. Siddiqui, J. S. Barbieri, A. L. Akhtar, W. Kim, R. Littman-Quinn, E. F. Conant, N. K. Gupta, B. A. Pukenas, P. Ramchandani *et al.*, "The accuracy of mobile teleradiology in the evaluation of chest x-rays," *Journal of telemedicine and telecare*, vol. 20, no. 8, pp. 460–463, 2014.

[86] M. Breuninger, B. van Ginneken, R. H. Philipsen, F. Mhimbira, J. J. Hella, F. Lwilla, J. van den Hombergh, A. Ross, L. Jugheli, D. Wagner *et al.*, "Diagnostic accuracy of computer-aided detection of pulmonary tuberculosis in chest radiographs: a validation study from sub-saharan africa," *PloS one*, vol. 9, no. 9, p. e106381, 2014.

[87] M. Muyoyeta, P. Maduskar, M. Moyo, N. Kasese, D. Milimo, R. Spooner, N. Kapata, L. Hogeweg, B. van Ginneken, and H. Ayles, "The sensitivity and specificity of using a computer aided diagnosis program for automatically scoring chest x-rays of presumptive tb patients compared with xpert mtb/rif in lusaka zambia," *PloS one*, vol. 9, no. 4, p. e93757, 2014.

[88] S. Jaeger, A. Karargyris, S. Candemir, L. Folio, J. Siegelman, F. Callaghan, Z. Xue, K. Palaniappan, R. K. Singh, S. Antani *et al.*, "Automatic tuberculosis screening using chest radiographs," *IEEE transactions on medical imaging*, vol. 33, no. 2, pp. 233–245, 2014.

[89] H. Müller and P. Clough, "Imageclef 2004–2005: results, experiences and new ideas for image retrieval evaluation," 2005.

[90] H. Müller, T. Deselaers, T. Deserno, P. Clough, E. Kim, and W. Hersh, "Overview of the imageclefmed 2006 medical retrieval and medical annotation tasks," in *Workshop of the Cross-Language Evaluation Forum for European Languages.* Springer, 2006, pp. 595–608.

[91] H. Müller, T. Deselaers, T. M. Deserno, J. Kalpathy-Cramer, E. Kim, and W. Hersh, "Overview of the imageclefmed 2007 medical retrieval and medical annotation tasks," in *Workshop of the Cross-Language Evaluation Forum for European Languages.* Springer, 2007, pp. 472–491.

[92] H. Müller, J. Kalpathy-Cramer, I. Eggel, S. Bedrick, S. Radhouani, B. Bakke, C. E. Kahn, and W. Hersh, "Overview of the clef 2009 medical image retrieval track," in *Workshop of the Cross-Language Evaluation Forum for European Languages.* Springer, 2009, pp. 72–84.

[93] H. Müller, J. Kalpathy-Cramer, C. E. Kahn, W. Hatt, S. Bedrick, and W. Hersh, "Overview of the imageclefmed 2008 medical image retrieval task," in *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, 2008, pp. 512–522.

[94] H. Müller, A. G. S. de Herrera, J. Kalpathy-Cramer, D. Demner-Fushman, S. K. Antani, and I. Eggel, "Overview of the imageclef 2012 medical image retrieval and classification tasks." in *CLEF (online working notes/labs/workshop)*, 2012, pp. 1–16.

[95] B. Hu, S. Dasmahapatra, P. Lewis, and N. Shadbolt, "Ontology-based medical image annotation with description logics," in *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 2003, pp. 77–82.

[96] D. L. Rubin, P. Mongkolwat, V. Kleper, K. Supekar, and D. S. Channin, "Medical imaging on the semantic web: Annotation and image markup." in *AAAI Spring Symposium: semantic scientific knowledge integration*, 2008, pp. 93–98.

[97] C. Liu, Y. Cao, M. Alcantara, B. Liu, M. Brunette, J. Peinado, and W. Curioso, "Tx-cnn: Detecting tuberculosis in chest x-ray images using convolutional neural network," in *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2314–2318.

[98] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[99] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[100] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[101] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 2013, pp. 411–418.

[102] H.-C. Shin, H. R. Roth, M. Gao, L. Lu *et al.*, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[103] M. F. Alcantara, Y. Cao, C. Liu, B. Liu *et al.*, "Improving tuberculosis diagnostics using deep learning and mobile health technologies among resource-poor communities in perú," *Smart Health*, vol. 1, pp. 66 – 76, 2017.

[104] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[105] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[106] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *2007 IEEE 11th international conference on computer vision*. Ieee, 2007, pp. 1–8.

[107] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[108] L. Shen, Z. Lin, and Q. Huang, "Relay backpropagation for effective learning of deep convolutional neural networks," *arXiv preprint arXiv:1512.05830*, 2015.

[109] M. L. Meuter, A. L. Ostrom, R. I. Roundtree, and M. J. Bitner, "Self-service technologies: understanding customer satisfaction with technology-based service encounters," *Journal of marketing*, vol. 64, no. 3, pp. 50–64, 2000.

[110] S. Bulmer, J. Elms, and S. Moore, "Exploring the adoption of self-service checkouts and the associated social obligations of shopping practices," *Journal of Retailing and Consumer Services*, vol. 42, pp. 107–116, 2018.

[111] D. Grewal, A. L. Roggeveen, and J. Nordfält, "The future of retailing," *Journal of Retailing*, vol. 93, no. 1, pp. 1–6, 2017.

[112] N. Wingfield, "Amazon moves to cut checkout line, promoting a grab-and-go experience," *The New York Times*, 2016.

[113] I. Krasin, T. Duerig, N. Alldrin, A. Veit, S. Abu-El-Haija, S. Belongie, D. Cai, Z. Feng, V. Ferrari, V. Gomes *et al.*, "Openimages: A public dataset for large-scale multi-label and multi-class image classification," *Dataset available from https://github. com/openimages*, vol. 2, no. 6, p. 7, 2016.

[114] S. Hou, Y. Feng, and Z. Wang, "Vegfru: A domain-specific dataset for fine-grained visual categorization," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 541–549.

[115] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 4510–4520.

[116] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Cachier: Edge-caching for recognition applications," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 276–286.

[117] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[118] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[119] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[120] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[121] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[122] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[123] C. Liu, X. Wang, J. Ni, Y. Cao, and B. Liu, "An edge computing visual system for vegetable categorization," *arXiv preprint*, 2019.

[124] S. R. Dubey and A. S. Jalal, "Fruit and vegetable recognition by fusing colour and texture features of the image using machine learning," *International Journal of Applied Pattern Recognition*, vol. 2, no. 2, pp. 160–181, 2015.

[125] K. Hameed, D. Chai, and A. Rassau, "A comprehensive review of fruit and vegetable classification techniques," *Image and Vision Computing*, vol. 80, pp. 24–44, 2018.

[126] Y. Sakai, T. Oda, M. Ikeda, and L. Barolli, "A vegetable category recognition system using deep neural network," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2016 10th International Conference on*. IEEE, 2016, pp. 189–192.

[127] H. Mureşan and M. Oltean, "Fruit recognition from images using deep learning," *Acta Universitatis Sapientiae, Informatica*, vol. 10, no. 1, pp. 26–42, 2018.

[128] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "Deepfruits: A fruit detection system using deep neural networks," *Sensors*, vol. 16, no. 8, p. 1222, 2016.

[129] Y. Kawano and K. Yanai, "Automatic expansion of a food image dataset leveraging existing categories with domain adaptation," in *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014.

[130] H. M. Zawbaa, M. Abbass, M. Hazman, and A. E. Hassenian, "Automatic fruit image recognition system based on shape and color features," in *International Conference on Advanced Machine Learning Technologies and Applications*. Springer, 2014, pp. 278–290.

[131] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[132] C. S. A. H. S. B. Yin Cui, Yang Song, "Large scale fine-grained categorization and domain-specific transfer learning," in *CVPR*, 2018.

[133] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[134] M. F. Alcantara, Y. Cao, C. Liu, B. Liu, M. Brunette, N. Zhang, T. Sun, P. Zhang, Q. Chen, Y. Li *et al.*, "Improving tuberculosis diagnostics using deep learning and mobile health technologies among resource-poor communities in perú," *Smart Health*, vol. 1, pp. 66–76, 2017.

BIOGRAPHICAL SKETCH

Chang Liu received his bachelor's degree in Computer Science and Technology from Huazhong University of Science and Technology in China, and master's degree in Computer Science from University of Massachusetts Lowell, USA. His research interests lie in machine learning, computer vision, edge computing and medical imaging. His publications include [18, 42, 61, 97, 134]