Diabetic Retinopathy Classification Using an Efficient Convolutional Neural Network

by

Jiaxi Gao

B.Eng., Xi'an Jiaotong University, 2015

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)

April 2019

© Jiaxi Gao, 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, a thesis/dissertation entitled:

DIABETIC RETINOPATHY CLASSIFICATION USING AN EFFICIENT CONVOLUTIONAL
NEURAL NETWORK

submitted by JIAXI GAO in partial fulfillment of the requirements for

the degree of MASTER OF APPLIED SCIENCE

in ELECTRICAL AND COMPUTER ENGINEERING

Examining Committee:

CYRIL LEUNG, ELECTRICAL AND COMPUTER ENGINEERING

Supervisor

RABAB WARD, ELECTRICAL AND COMPUTER ENGINEERING

Supervisory Committee Member

JOSÉ MARTÍ, ELECTRICAL AND COMPUTER ENGINEERING

Supervisory Committee Member

Abstract

Diabetic Retinopathy (DR) is a diabetic complication that affects the eyes and may lead to blurred vision or even blindness. The diagnosis of DR through retinal fundus images is traditionally performed by ophthalmologists who inspect for the presence and significance of many subtle features, a process which is cumbersome and time-consuming. As there are many undiagnosed and untreated cases of DR, DR screening of all diabetic patients is a huge challenge.

Deep convolutional neural network (CNN) has rapidly become a powerful tool for analyzing medical images. There have been previous works which use deep learning models to detect DR automatically. However, these methods employed very deep CNNs which require vast computational resources. Thus, there is a need for more computationally efficient deep learning models for automatic DR diagnosis. The primary objective of this research is to develop a robust and computationally efficient deep learning model to diagnose DR automatically.

In the first part of this thesis, we propose a computationally efficient deep CNN model MobileNet-Dense which is based on the recently proposed MobileNetV2 and DenseNet models. The effectiveness of the proposed MobileNet-Dense model is demonstrated using two widely used benchmark datasets, CIFAR-10 and CIFAR-100.

In the second part of the thesis, we propose an automatic DR classification system based on the ensemble of the proposed MobileNet-Dense model and the MobileNetV2 model. The performance of our system is evaluated and compared with

some of the state-of-the-art methods using two independent DR datasets, the EyePACS dataset and the Messidor database. On the EyePACS dataset, our system achieves a quadratic weighted kappa (QWK) score of 0.852 compared to a QWK score of 0.849 achieved by the benchmark method while using 32% fewer parameters and 73% fewer multiply-adds (MAdds). On the Messidor database, our system outperforms the state-of-the-art method on both Normal/Abnormal and Referable/Non-Referable classification tasks.

Lay Summary

Diabetic Retinopathy (DR) is a diabetic complication that affects the eyes and may cause vision impairment or even vision loss. The presence of the disease will result in progressively developing abnormalities such as exudates, hemorrhages and microaneurysms in the retina. As the number of the patient with undiagnosed DR is increasing globally and manually diagnosing DR is cumbersome and time-consuming, performing DR screening automatically has a great significance.

Convolutional Neural Network (CNN) has rapidly become a popular tool for medical image processing and analysis. Previous works which applied deep CNN models on the automatic screening of DR needs vast computational resources. In this thesis, we study the use of the computationally efficient deep CNN model for the automatic classification of DR.

Preface

I hereby declare that I am the author of this thesis. This thesis is an original, unpublished work carried out under the supervision of Professor Cyril Leung.

Table of Contents

Abstract	iii
Lay Summary	v
Preface	vi
Table of Contents	vii
List of Tables	xi
List of Figures	xii
List of Abbreviations	xv
Notation	xvii
Acknowledgements	xviii
Dedication	xix
1 Introduction	1
1.1 An Overview of Diabetic Retinopathy	1
1.1.1 Diabetes Mellitus and Diabetic Retinopathy	1
1.1.2 Diabetic Retinopathy Classification Rules	2
1.2 Related Work	3

Table of Contents

1.2.1	DR Screening Using Traditional Machine Learning Algorithms	4
1.2.2	DR Screening Using Deep Learning Algorithms	6
1.2.3	Discussions	8
1.3	Motivations	8
1.4	Contributions	9
1.5	Organization of the Thesis	10
2	Review of Neural Network Models	12
2.1	Fully Connected Neural Networks	12
2.1.1	The Architecture of FCNN	12
2.1.2	Training the Neural Network Model	13
2.2	Convolutional Neural Networks	17
2.2.1	Standard Convolutional Layer	18
2.2.2	Depthwise Separable Convolutional Layer	22
2.2.3	Max Pooling Layer	28
2.2.4	Fully Connected Layer	29
2.3	Typical Workflow of a Deep Learning Project	30
3	MobileNet-Dense: An Efficient Convolutional Neural Network	33
3.1	Related Works	33
3.1.1	CNN with Skip Connections: ResNet and DenseNet	33
3.1.2	Efficient CNNs: MobileNetV1 and MobileNetV2	38
3.2	MobileNet-Dense: An efficient CNN based on Dense Connections and Depthwise Separable Convolutional Layers	45
3.3	Experiment	47
3.3.1	Experiment Environment	47
3.3.2	CIFAR Datasets	49

Table of Contents

3.3.3	Training	49
3.3.4	Performance Evaluation	50
3.4	Summary	52
4	Automatic Diabetic Retinopathy Classification System Design	53
4.1	Exploratory Data Analysis	53
4.1.1	EDA of Training Labels	55
4.1.2	EDA of Training Images	55
4.2	Image Augmentation and Preprocessing	58
4.3	CNN Training	61
4.3.1	Evaluation Metric and Loss Function	61
4.3.2	Hyperparameter Tuning and Training	65
4.4	Ensemble Learning	67
4.4.1	Feature Extraction and Feature Blending	68
4.4.2	Feature Reduction	68
4.4.3	Model Stacking	69
4.5	Summary	69
5	Performance Evaluation of the Diabetic Retinopathy Classification System	70
5.1	Performance Evaluation on the EyePACS Test Set	70
5.1.1	The QWK score and Model Complexity	71
5.1.2	Performance Evaluation for Each Class	72
5.2	Performance Evaluation on the Messidor Database	75
5.2.1	EDA of the Messidor Database	75
5.2.2	Performance Evaluation	76
5.3	Summary	81

Table of Contents

6 Conclusion	83
6.1 Main Contributions	83
6.2 Future Work	84
Bibliography	86

List of Tables

1.1	DR grading rules [1]	4
3.1	Error rates of plain CNNs and ResNets on ImageNet dataset [2]	34
3.2	The structure of MobileNetV1 [3]	41
3.3	The structure of MobileNetV2 [4]	44
3.4	The structure of MobileNet-Dense	48
3.5	The structure of MobileNetV2 for CIFAR datasets	51
3.6	The structure of MobileNet-Dense for CIFAR datasets	51
3.7	Performance on CIFAR-10 and CIFAR-100 datasets	52
4.1	DR label information in EyePACS dataset [5]	55
4.2	The structure of MobileNet-Dense for DR classification	66
4.3	The structure of MobileNetV2 for DR classification	66
5.1	Performance comparison on EyePACS test set	71
5.2	Per-class performance on EyePACS test set	75
5.3	Performance comparison on Normal/Abnormal screening task	81
5.4	Performance comparison on Referable/Non-Referable screening task	81

List of Figures

1.1	Annotated results of an image with DR [6].	3
2.1	The structure of a FCNN with two hidden layers.	13
2.2	The structure of the AlexNet [7].	18
2.3	Illustration of convolving a single $7 \times 7 \times 3$ filter over a $32 \times 32 \times 3$ input feature map with a stride of 1 (shifting the filter one unit horizontally or vertically at a time). There are 26×26 spatial locations for a $7 \times 7 \times 3$ filter to slide over a $32 \times 32 \times 3$ input feature map, thus this procedure generates a $26 \times 26 \times 1$ output feature map, where each element in the output feature map is the sum of the element-wise multiplication of the filter and the small patch of the input feature map it overlapped.	23
2.4	Illustration of convolving a $D_F \times D_F \times M$ input feature map with a standard convolutional layer. This convolutional layer contains N convolution filters. Each filter has a dimension of $D_K \times D_K \times M$. .	24
2.5	Illustration of convolving a $D_F \times D_F \times M$ input feature map with a depthwise convolutional layer. The depthwise convolutional layer contains M convolutional filters. Each filter has a dimension of $D_K \times D_K \times 1$	25

2.6	Illustration of convolving a $D_G \times D_G \times M$ input feature map with a pointwise convolutional layer. The pointwise convolutional layer contains N convolutional filters. Each filter has a dimension of $1 \times 1 \times M$	27
2.7	Illustration of maxpooling operations.	29
3.1	Building blocks of ResNet [2].	36
3.2	Building blocks of DenseNet [8].	37
3.3	Illustration of a typical Dense Block [8] constructed using 5 bottleneck blocks. Each bottleneck block is illustrated by a single blue node . Any two of the nodes within the same dense block are directly connected.	39
3.4	A deep DenseNet [8] with two dense blocks. The Conv1x1 layer and pooling layer between two dense blocks are applied to downsample the size of the feature map.	40
3.5	Building block of MobileNetV1 [3].	41
3.6	Building blocks of MobileNetV2 [4].	43
3.7	Building blocks of MobileNet-Dense	46
4.1	Flowchart of the proposed automatic DR classification system.	54
4.2	Distribution of DR label values in EyePACS training set.	56
4.3	Label correlation in the EyePACS training set.	57
4.4	Random sample images from the EyePACS dataset [5].	58
4.5	Underexposed/Normal/Overexposed Images. Small figures with Red/Green/Blue color under each retinal image are the histograms of the Red/Green/Blue color channel.	59
4.6	Illustration of the original image and the augmented image using Fancy PCA algorithm.	61

List of Figures

5.1	Confusion matrices of: (a) MobileNetV2, (b) MobileNet-Dense, (c) Model Ensemble (2+1)	74
5.2	Random sample images from the Messidor database [9].	76
5.3	Distribution of DR grades in the Messidor Database.	77
5.4	ROC curves for Normal/Abnormal task.	79
5.5	ROC curves for Referable/Non-Referable task.	80

List of Abbreviations

ANN	Artificial Neural Network
Acc	Accuracy
AUC	Area Under ROC Curve
CNN	Convolutional Neural Network
Conv	Convolutional Layer
DM	Diabetes Mellitus
DR	Diabetic Retinopathy
EDA	Exploratory Data Analysis
FCNN	Fully Connected Neural Network
FN	False Negative
FP	False Positive
MAdds	Multiply-adds
MSE	Mean Squared Error
NPDR	Non-Proliferative Diabetic Retinopathy
PCA	Principal Component Analysis
PDR	Proliferative Diabetic Retinopathy
QWK	Quadratic Weighted Kappa
Relu	Rectified Linear Unit
ROC	Receiver Operating Characteristic Curve
SD	Standard Deviation
TN	True Negative

List of Abbreviations

TP	True Positive
----	---------------

Notation

$\mathbf{A}$	Matrix
$a_{i,j}$	ij^{th} entry of Matrix $\mathbf{A}$
$\mathbf{a}$	Vector
$\mathbf{1}$	All-one column vector
$\mathbf{I}$	Identity matrix
$(\cdot)^T$	Transpose
$*$	Convolution operation
$\star$	Cross-correlation operation
$\text{var}(\cdot)$	Variance operator
$\text{Cov}(\cdot)$	Covariance operator
$\sigma(\cdot)$	Non-linear activation function
$\odot$	Hadamard product

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Cyril Leung, for his immeasurable support and guidance throughout my research studies. His patience and guidance help me overcome challenges and finish this thesis. Without Professor Leung's guidance, this thesis would not have been possible.

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Grant RGPIN 1731-2013, the UBC Faculty of Applied Science, the UBC PMC-Sierra Professorship in Networking and Communications and the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative.

I would like to thank to EyePACS and Messidor program partners for providing the retinal image dataset.

I would like to thank my co-supervisor, Professor Chunyan Miao (Nanyang Technological University), for her guidance and assistance during my studies in Singapore. I would also like to thank all colleagues who helped me with my research.

Most importantly, none of this would have been possible without the love, encouragement and patience of my parents. I am deeply grateful to my parents who encourage me to pursue further education abroad. Without my parents, I would not be who I am today.

Dedication

To my parents and families.

Chapter 1

Introduction

This chapter provides a brief introduction to diabetic retinopathy (DR). In Section 1.1, we discuss DR, the importance of diagnosing DR and the diagnosis rules. Section 1.2 includes a literature review of previous works on the automatic diagnosis of DR using machine learning approaches. The motivation and contributions of this thesis are discussed and outlined in Section 1.3. The structure of this thesis is outlined in Section 1.4.

1.1 An Overview of Diabetic Retinopathy

1.1.1 Diabetes Mellitus and Diabetic Retinopathy

Diabetes Mellitus (DM) is a chronic disease characterized by consistently high blood glucose levels over a long-term period [10]. Without controlling blood glucose levels, many long-term complications may occur such as diabetic retinopathy (DR), diabetic foot or diabetic kidney disease [11].

DR is a chronic and progressive diabetic complication which damages the retina. Globally, DR will affect 191 million people by 2030 [12]. DR is caused by lasting harm to the retinal vessels from the high blood glucose level which can lead to blockage or damage in the tiny retinal blood vessels which nourish the retina. In response, the human body attempts to grow new blood vessels in the eye to maintain the nourishment. The new blood vessels are weak, which makes them have a high probability of leaking and bleeding [13]. As a result, patients may experience progressive vision

disorders from blurred vision to vision loss [14].

1.1.2 Diabetic Retinopathy Classification Rules

According to [14], there are 2 stages in the DR disease: Non-Proliferative Diabetic Retinopathy (NPDR) and Proliferative Diabetic Retinopathy (PDR). For NPDR, the disease can be further categorized as mild NPDR, moderate NPDR, or severe NPDR [15]. NPDR is the early stage of DR during which the retinal arteries become weak and small red dots like microaneurysms or even hemorrhages can be found in the retinal images. PDR is a condition in which the retina lacks oxygen and there are spots that appear on the retina as a result of the circulatory system attempting to maintain the delivery of oxygen.

The classification rules for DR are given in Table 1.1. A sample retinal image with the manually annotated common DR lesions [6] is shown in Figure 1.1.

The typical lesions of DR are briefly discussed below:

- **Hard Exudates:** Hard exudates are one of the constellations of retinal lesions that define DR. Hard exudates usually appear in the retinal images as tiny yellow-white flecks with sharp edges and different sizes [16].
- **Soft Exudates:** Soft exudates, also known as cotton wool spots, appear as white patches with blurred and hazy edges [17]. Exudates, including soft exudates and hard exudates, are one of the most common early lesions of DR [18].
- **Microaneurysms:** Microaneurysms are the earliest clinically visible signs of NPDR and are caused by dilatations of thin blood vessels. Microaneurysms usually appear as small red dots with sharp edges (20 to 200 microns) in clusters [19].

Figure 1.1: Annotated results of an image with DR [6].

- **Haemorrhages:** Retinal hemorrhages are the bleeding spots in the retina. It appears like a blot in the retinal fundus image and its shape varies [17].

1.2 Related Work

In this section, we review and discuss some previous works on the automatic detection of DR.

Currently, DR is mostly diagnosed manually by inspecting retinal images. The process is time-consuming and challenging since some lesions in the retinal image are very tiny or subtle, such as microaneurysm illustrated in Figure 1.1. In order to improve the efficiency and accuracy of DR classifications, many automatic or semi-automatic methods using computer vision and machine learning algorithms have been

Table 1.1: DR grading rules [1]

Disease Severity Level	Findings
Grade 0: No apparent retinopathy	No visible sign of abnormalities
Grade 1: Mild NPDR	Presence of microaneurysms only
Grade 2: Moderate NPDR	More than just microaneurysms but less than severe NPDR
Grade 3: Severe NPDR	Any of the following: (1) More than 20 intraretinal hemorrhages (2) Venous beading (3) Intraretinal microvascular abnormalities (4) No signs of PDR
Grade 4: PDR	Either or both of the following: (1) Neovascularization (2) Vitreous/pre-tinal hemorrhage

proposed.

Generally, automatic DR screening algorithms have evolved from traditional computer vision techniques which combined manually designed feature extraction algorithms and traditional classification algorithms to end-to-end deep learning algorithms.

1.2.1 DR Screening Using Traditional Machine Learning Algorithms

Before the development of deep learning algorithms, especially deep neural networks, a feature extraction step was needed for general computer vision tasks. Features, in this case, are distinguishing and significative small image patches. Many feature extraction algorithms were proposed in the 1990s, such as SIFT [20] and SURF [21], which have been widely applied for object recognition [22] or medical image retrieval [23]. Following the feature extraction step, traditional machine learning classification algorithms, such as support vector machines, logistic regression or decision trees, are trained using extracted features to classify the image. This pipeline was applied to almost all traditional computer vision tasks including image classification, image

segmentation, etc.

Automatic DR detection based on traditional computer vision techniques follows the same pipeline. First, specific features are extracted from the fundus images using single or combinations of manually designed feature extraction algorithms. Classifiers are then trained using the extracted feature to classify the DR stages. Several such works are briefly reviewed below.

A multilayer neural network model was trained using features extracted by recursive region growing segmentation algorithms to perform binary DR classification task [24]. The sensitivity and specificity of the proposed system is 80.2% and 70.7% respectively.

Recursive region growing segmentation algorithms was also adopted in [25] to extract DR visual features such as exudates, hemorrhages, and microaneurysms. The features were then used for Healthy/DR binary classification. The sensitivity and specificity of the proposed system is 74.8% and 82.7% respectively.

Lee et al. [26] introduced methods for automatically grading the severity of three early DR lesions, namely hemorrhages, hard exudates, and cotton-wool spots. An algorithm is trained to classify NPDR based on the absence of these three types of lesions. The overall accuracy of the proposed system is 81.7%.

In [27], a decision support system was proposed for early detection of DR (presence of microaneurysms) using Bayes optimality criteria. The method was able to identify the early stage of DR with a sensitivity of 100% and specificity of 67%.

The automatic DR classification methods discussed in this section use manually designed features. The performance of such methods is highly dependent on the feature extraction process. Machine learning algorithms are only used for numerical optimization based on the features designed by humans. Thus, domain knowledge

of the DR disease plays a crucial role in building effective DR classification models. The limitation of applying conventional computer vision algorithms for DR lesion detection or DR classification is that the manually designed features are often over-specified, incomplete or require a long time and experience to design and validate.

1.2.2 DR Screening Using Deep Learning Algorithms

Instead of manually designing features and feeding them to the classifier for DR screening, many researchers are now building end-to-end deep learning models which learn all the needed features automatically. Several works which use deep neural network models to detect DR automatically are briefly reviewed below.

A 13-layer deep convolutional neural network (CNN) for screening DR was discussed in [29]. Several data augmentation techniques (i.e., image rotation, image flipping and image shifting) were applied to overcome the data imbalanced problem. Five thousand images from the EyePACS DR training set [5] were retained as the test set and the rest of the images from the EyePACS DR training set were used for training the proposed model. The trained network achieves 95% specificity and 30% sensitivity on the test set, which indicates that the proposed model is highly biased to the negative (healthy) case.

In [30], a residual neural network [2] was used to classify the retinal image. The quadratic weighted kappa (QWK) [31] was adopted as the evaluation metric. The proposed system achieves a QWK score of 0.51 on the EyePACS DR test set.

A deeply supervised ResNet was proposed in [32] to automatically classify the grade of DR. In the proposed architecture, three sets of additional side-output layers were added to the intermediate layers of the ResNet to provide additional regularization during the training process. 20% images from the EyePACS DR training dataset were retained as the test set. The results show that by leveraging the predictions of

intermediate supervised layers, ResNet can focus on multi-scale learning which leads to improved performance compared to standard ResNet without side-output layers. The QWK score, specificity, sensitivity and accuracy of the proposed system on the test set is 0.73, 94%, 67% and 81% respectively.

In [33], a Non-Local means denoising method was applied to the fundus image to remove noise from the EyePACS DR dataset. AlexNet [7] and GoogleNet[34] were utilized to classify the pre-processed retinal images. An AUC score of 0.78 is achieved by the GoogleNet model compared to an AUC score of 0.68 achieved by AlexNet.

In [35], a computationally efficient CNN model, namely the MobileNet model [3], was used for binary DR screening (DR V.S. Healthy image) using the EyePACS DR dataset. An accuracy of 0.73 was achieved by the proposed method. Considering the great data imbalance nature of this dataset (73.4% of the image are healthy and 26.6% of the image contains DR), only using accuracy score may not appropriate for demonstrating the effectiveness of the proposed model. For an extreme example, a simple classifier which always predicts images as healthy images will also achieve an accuracy of 0.73 using the EyePACS DR dataset.

In [36], GoogleNet and VGGNet were modified to two corresponding networks CKML (Combined Kernels with Multiple Losses Network) and VNXX (VGGNet with Extra Kernel). The retinal images were converted to a hybrid LGI color space. Models trained using the LGI retinal images were then compared with models trained using RGB retinal images. The results show that both CKML and VNXX achieve higher accuracy with LGI images. However, the proposed model is also highly biased to the healthy samples. The corresponding accuracy from class 0(healthy) to class 4(proliferative DR) are 97.6%,11.9%,57.9%,33.2% and 36.8% respectively.

In [37], a novel Zoom-in-Net model was proposed. It mimics the zoom-in process

of an ophthalmologist who examines the retinal images. Zoom-in-Net consists of three parts, Inception-ResNet was adopted as the main network (M-Net) for DR classification. Two tiny CNNs, namely the Attention Network (A-Net) and the Crop Network (C-Net), were used for Attention Localization of the for DR images. The single Zoom-in-Net achieves a QWK score of 0.849 on the entire EyePACS DR test set. With an ensemble of three Zoom-in-Net models, the QWK score was improved to 0.854.

1.2.3 Discussions

There are few comparisons of the discussed deep learning methods although all of the methods used the EyePACS DR dataset. Possible reasons include:

1. Different classification tasks were performed. For example, binary DR classification (Healthy vs DR) task was performed in [33, 35], while DR classification with 5 classes was considered in [36, 37, 32].
2. Different number of retinal images were retained as the test set. For example, 20% of the retinal images in the EyePACS DR test set were used to report the model performance in [32, 29]. The entire EyePACS DR test set was used to report the model performance in [37].
3. Different classification metrics were used to assess performance. For example, accuracy (Acc) score was used in [38, 33] while quadratic weighted kappa (QWK) score was reported in [30, 32, 37].

1.3 Motivations

DR is one of the main causes of vision loss and blindness. Manually diagnosing DR from retinal images is time-consuming and challenging. To this end, several works

have applied deep CNN models to diagnose DR automatically. However, the proposed methods employed very deep CNN models (e.g., ResNet-based method [32], InceptionNet-based method [33], InceptionNet-ResNet-based model [37]) which require vast computational resources. Research on DR screening using *computationally efficient CNN models* has a great practical significance. This thesis is therefore focused on applying end-to-end, accurate and computationally efficient CNN models for automatic DR classification.

1.4 Contributions

The main contributions of this thesis are summarized as follows:

- In Chapter 3, a computationally efficient CNN model MobileNet-Dense is proposed. Two benchmark datasets, namely CIFAR-10 and CIFAR-100, are used to demonstrate the effectiveness of the proposed MobileNet-Dense model. The accuracy (Acc) scores of MobileNet-Dense are compared with the Acc scores of MobileNetV2 model [4] on these two datasets. The MobileNet-Dense model and MobileNetV2 model achieve similar Acc scores on both CIFAR-10 and CIFAR-100 datasets, but the proposed MobileNet-Dense uses 62% fewer parameters and 36% fewer multiply-adds (MAdds).
- In Chapter 4, a computationally efficient automatic DR classification system is proposed. The proposed system is based on the ensemble of the proposed MobileNet-Dense model and the MobileNetV2 model.
- In Chapter 5, the performance of the proposed DR classification system is evaluated using the EyePACS DR dataset and the Messidor database. For EyePACS DR dataset, the top-ranked method [39] in the Kaggle DR challenge

is chosen as the benchmark method. Our system achieves a QWK score of 0.852 compared to a QWK score of 0.849 achieved by the benchmark method while using 32% fewer parameters and 73% fewer MAdds. For the Messidor database, the proposed system is evaluated on Normal/Abnormal screening task and Referable/Non-Referable screening task. On Normal/Abnormal screening task, our system achieves an AUC of 0.962 and an Acc of 0.917 compared to an AUC of 0.921 and an Acc of 0.905 achieved by the state-of-the-art method [37]. On Referable/Non-Referable screening task, our system achieves an AUC of 0.970 and an Acc of 0.924 compared to an AUC of 0.957 and an Acc of 0.911 achieved by the state-of-the-art method [37].

1.5 Organization of the Thesis

The remainder of the thesis is organized as follows:

In Chapter 2, we introduce some basic concepts of deep learning methods which are directly related to the work described in later chapters. Specifically, we first briefly review the basic concepts and core layers of conventional fully connected neural networks (FCNN). Then we recall the basic concepts and core layers of convolutional neural networks (CNNs), including the standard convolutional layers, the depthwise separable convolutional layers, the pooling layers and the fully connected layers. The typical workflow of applying a deep learning model for a real-world task is summarized in the last section.

In Chapter 3, we propose a computationally efficient MobileNet-Dense model and we demonstrate its effectiveness using two benchmark datasets, the CIFAR-10 and CIFAR-100 datasets.

In Chapter 4, we propose a computationally efficient DR classification system

based on the ensemble of the proposed MobileNet-Dense model and the MobileNetV2 model. The system is trained using the publicly available EyePACS DR dataset. The workflow of the proposed system is described in details.

In Chapter 5, we demonstrate the effectiveness and efficiency of the proposed automatic DR classification system using the EyePACS DR dataset and the Messidor database.

In Chapter 6, the main contributions of this thesis are summarized and some possible future extensions are outlined.

Chapter 2

Review of Neural Network Models

The primary approach in this research project is to apply *computationally efficient CNNs* for automatic diagnosis of DR. This chapter provides some background on neural network models especially on CNNs and computationally efficient CNNs. In Section 2.1, we review the conventional fully connected neural network which is the foundation of CNNs. In Section 2.2, we review the core layers of CNNs, including the standard convolutional layer, the depthwise separable convolutional layer, the maxpooling layer and the fully connected layer. The typical workflow of applying neural network models for a real-world task is summarized in Section 2.3.

2.1 Fully Connected Neural Networks

A fully connected neural network (FCNN) is a neural network model inspired by biological neural networks [40].

2.1.1 The Architecture of FCNN

The structure of a simple FCNN with two hidden layers is shown in Figure 2.1. There are three types of layers in a typical FCNN. The input layer, the hidden layer and the output layer. The input layer is used to take in the input information. The hidden layers are used to perform a non-linear combination of the information from the previous (input or hidden) layer. The output layer performs a weighted sum on the results of the last hidden layer. All layers are composed of neurons (also known as nodes). Nodes in two adjacent layers are fully-connected by edges. Each edge

Figure 2.1: The structure of a FCNN with two hidden layers.

represents a weight value learned during the training process.

2.1.2 Training the Neural Network Model

Training a neural network model consists of executing the feed forward process and the backpropagation process for all training samples iteratively.

The Feed Forward Process

The feed forward process refers to feeding the input vector into a layer or a neural network model to calculate the output vector.

Let $w_{j,k}^l$ denotes the weight value between the k^{th} node in the $(l-1)^{th}$ layer and the j^{th} node in the l^{th} layer. b_j^l denotes the bias value of the j^{th} node in the l^{th} layer and z_j^l denotes the output value (before activation) of the j^{th} node in the l^{th} layer. Let $\sigma(\cdot)$ denotes the non-linear activation function applied on z_j^l . a_j^l denotes

the activation of z_j^l . There are several types of activation functions (e.g., Sigmoid $f(x) = \frac{1}{1+\exp(-x)}$, Tanh $f(x) = \frac{2}{1+\exp(-2x)} - 1$ or Relu $f(x) = \max(0, x)$). For training deep neural networks, the Relu (Rectified Linear Unit) activation function is preferred since it does not have the vanishing gradient problem [41] and it leads to more sparsity which accelerates the training process [42]. The feed forward process from $(l-1)^{th}$ layer to l^{th} layer is given by:

$$a_j^l = \sigma(z_j^l) = \sigma\left(\sum_k w_{j,k}^l a_k^{l-1} + b_j^l\right) \quad (2.1)$$

We can rewrite Eq. (2.1) in the vector form as follows:

$$\mathbf{a}^l = \sigma(\mathbf{z}^l) = \sigma(\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l) \quad (2.2)$$

where $\mathbf{W}^l$ denote the weight matrix from the $(l-1)^{th}$ layer to the l^{th} layer. $\mathbf{a}^l$ and $\mathbf{b}^l$ are column vectors which denote the activation and bias of the l^{th} layer respectively. Suppose there are L layers in the neural network model. Given an input vector $\mathbf{a}^0$, we can calculate the output of l^{th} ($0 < l < L$) layer $\mathbf{a}^l$ using Eq. (2.2).

The Backpropagation Process

The backpropagation algorithm [43] is used for updating the learnable parameters (i.e., weights and bias) during backpropagation process, in such a way as to minimize the loss function. The loss functions are used to measure the error (disagreement) between the predicted results and the ground truth label and the error is propagated backward throughout the neural network's layers (i.e., the error is propagated from the output layer to the input layer). There are many loss functions which are commonly used, e.g., categorical cross-entropy loss or mean squared error (MSE). The categorical cross-entropy loss is often used for classification task and is given by:

$$\mathcal{L} = \frac{1}{N} \sum_i \sum_j t_{i,j} \log(p_{i,j}). \quad (2.3)$$

where $p_{i,j}$ denotes the predicted probability value of sample i to be in class j given by the neural network model, $t_{i,j}$ denotes the ground truth probability value of sample i to be in class j (i.e., $t_{i,j} = [0, \dots, 1, \dots, 0]$ contains a single 1 at the j^{th} position if sample i is labeled as class j .) and N denotes the total number of samples.

The MSE loss is often used for regression task and is given by:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (o_i - t_i)^2. \quad (2.4)$$

where N denotes the total number of samples, o_i denotes the predicted value of sample i and t_i denotes the target value of sample i .

After an appropriate loss function $\mathcal{L}$ is chosen, the partial derivatives $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^l}$ are calculated using the chain rule to update the corresponding weights $\mathbf{W}^l$ and bias $\mathbf{b}^l$. To explain the backpropagation process, suppose we use $\mathbf{e}^l$ to denote the error of the output of the l^{th} layer. Then the $\mathbf{e}^l$ is given by:

$$\mathbf{e}^l = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^l} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^l} \frac{\partial \mathbf{a}^l}{\partial \mathbf{z}^l} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^l} \odot \sigma'(\mathbf{z}^l) \quad (2.5)$$

where the $\odot$ denotes the Hadamard product [44]. Using the chain rule, $\mathbf{e}^l$ is calculated using $\mathbf{e}^{l+1}$, as shown below:

$$\begin{aligned} \mathbf{e}^l &= \frac{\partial \mathcal{L}}{\partial \mathbf{a}^l} \odot \sigma'(\mathbf{z}^l) \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{l+1}} \frac{\partial \mathbf{z}^{l+1}}{\partial \mathbf{a}^l} \odot \sigma'(\mathbf{z}^l) \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{l+1}} \frac{\partial (\mathbf{W}^{l+1} \mathbf{a}^l + \mathbf{b}^{l+1})}{\partial \mathbf{a}^l} \odot \sigma'(\mathbf{z}^l) \\ &= ((\mathbf{W}^{l+1})^T \mathbf{e}^{l+1}) \odot \sigma'(\mathbf{z}^l) \end{aligned} \quad (2.6)$$

Suppose there are L layers in the neural network model. Given the error of the output layer $\mathbf{e}^L$ (measured using loss function), the error of the l^{th} layer $\mathbf{e}^l$ can be calculated using Eq. (2.6).

The partial derivatives of the loss function with respect to $\mathbf{W}^l$ and $\mathbf{b}^l$ are calculated using $\mathbf{e}^l$, as shown below:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^l} \frac{\partial \mathbf{z}^l}{\partial \mathbf{W}^l} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^l} \frac{\partial (\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l)}{\partial \mathbf{W}^l} = \mathbf{e}^l (\mathbf{a}^{l-1})^T \quad (2.7)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^l} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^l} \frac{\partial \mathbf{z}^l}{\partial \mathbf{b}^l} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^l} \frac{\partial (\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l)}{\partial \mathbf{b}^l} = \mathbf{e}^l \quad (2.8)$$

The weights $\mathbf{W}^l$ and bias $\mathbf{b}^l$ are updated using the corresponding partial derivatives:

$$\mathbf{W}^l = \mathbf{W}^l - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}^l} \quad (2.9)$$

$$\mathbf{b}^l = \mathbf{b}^l - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{b}^l} \quad (2.10)$$

where α is the step size (also known as the learning rate). If α is too large, we may overshoot the optimal value. On the other hand, if α is too small, it will take a long time to reach the optimal value or the loss value may converge to a local minimum or saddle point instead of the global minimum [45]. Thus the optimal learning rate α need to be carefully searched.

The optimal learnable parameters are obtained by iteratively executing the feed-forward process and the backpropagation process until the loss converge. The epoch describes the number of times the model trained using the entire training set. In other words, once we have updated the learnable parameters using all the training

samples, an epoch has completed. In general, training a neural network model requires multiple epochs. Ideally, we should calculate the mean loss for all training samples and update the learnable parameters once using the gradients of the mean loss. In practice, the datasets we use may contain thousands or millions of samples (e.g., ImageNet has more than 1 million training images) and updating the learnable parameters with the mean loss of all training samples are computationally inefficient. A commonly used strategy to solve this problem is estimating the gradient and updating the parameters using a small batch of training samples at a time. In other words, the parameters of the neural network get many approximate updates instead of a more accurate single update for one epoch. For a concrete example, the learnable parameters will update 10 times for one epoch if we have a training set with 1000 samples and we select the mini-batch size to be 100. This mini-batch strategy works well in most practical applications. The pseudocode of the backpropagation algorithm with mini-batch is listed in Algorithm 2.1.

Algorithm 2.1 The Backpropagation Algorithm

1. **Sample** a mini-batch of data from training dataset, without replacement.
 2. **Forward** propagate the mini-batch samples through the neural networks.
 3. **Estimate** the partial derivatives with backpropagation.
 4. **Update** the learnable parameters using the partial derivatives.
 5. **Repeat** step 1 to step 4 until all the training samples are sampled (one epoch finished).
 6. **Repeat** step 5 until the loss value converges.
-

2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) is one of the most widely adopted deep neural network models for the current research literature. CNNs have been shown to perform very well in many computer vision tasks such as face recognition [46] or object

detection [47].

Figure 2.2 illustrates the architecture of AlexNet [7]. AlexNet is a typical CNN model which shed the lights on the design of the CNN architecture. AlexNet is the winner of the ImageNet Large Scale Visual Recognition Competition 2012 (ImageNet Competition is a benchmark competition for image recognition. There are over 14,000,000 images which categorized into 1000 classes in this dataset). AlexNet is the first CNN used in this competition and yields a reduction in the error rate from 25.8% to 16.4%. For the AlexNet architecture, there are three types of layers: standard convolutional layers, max pooling layers, and fully connected layers.

Figure 2.2: The structure of the AlexNet [7].

2.2.1 Standard Convolutional Layer

The convolutional layer is the most important layer in a CNN model and it is trained using the backpropagation algorithm to extract specific features from the input vector. The output of the convolutional layer, also known as the output feature map, are generated by convolving multiple filters over the input feature map. For convolution operations in CNNs, we often use convolutions over two axes (width and height). The convolution of a two-dimensional image $\mathbf{I}$ and a two-dimensional kernel $\mathbf{K}$ is given as follows [48]:

$$S(i, j) = (\mathbf{I} * \mathbf{K})(i, j) = \sum_m \sum_n I_{m,n} K_{i-m, j-n} \quad (2.11)$$

where m and n are the width and height of the convolution filter respectively. Normally we apply kernel with symmetric size (i.e., $m = n$). Using the commutative property of the convolution operation, we can rewrite Eq. (2.11) as:

$$S(i, j) = (\mathbf{K} * \mathbf{I})(i, j) = \sum_m \sum_n I_{i-m, j-n} K_{m,n} \quad (2.12)$$

In practice, many neural network libraries (e.g., Tensorflow [49]) implement a related function called cross-correlation function instead of implementing the convolution function. The convolutional layer is constructed using as many cross-correlation filters as desired. The definition of the cross-correlation function is given as follows [48]:

$$S_{i,j} = (\mathbf{K} \star \mathbf{I})(i, j) = \sum_m \sum_n I_{i+m, j+n} K_{m,n} \quad (2.13)$$

CNN with cross-correlation kernel and CNN with convolution kernel will give the same results as the backpropagation algorithm will learn the appropriate values of the kernel in the appropriate place. Given the definition of the convolution and the cross-correlation, the kernel implementation based on convolution will learn a kernel which is flipped relative to the kernel implementation based on cross-correlation [48]. In this thesis, the cross-correlation operation is referred to as the convolution operation.

Convolution operation preserves the spatial information between pixels by learning features only using small patches of the input feature map. Figure 2.3 illustrates the process of convolving a $32 \times 32 \times 3$ input feature map with a $7 \times 7 \times 3$ convolution filter (also known as kernel). The kernel slides across the input feature map over two axes (width and height). At each location, the element-wise multiplication of

the kernel value and the input element it overlapped is computed and the results are summed up to obtain the value at the current location for the output feature map.

More generally, a convolutional layer contains multiple convolution filters and the input feature map is convolved with each filter independently and thus each filter produces its own output feature map. The output of the convolutional layer is the stack of those output feature maps. The standard convolutional layer is illustrated in Figure 2.4. We use the same notations used in [3] to describe the convolutional layer. To be more specific, D_F, D_K and D_G denote the spatial width (height) of the input feature map, the kernel and the output feature map respectively. The M and N denote the depth of the input feature map and the output feature map respectively. In this standard convolutional layer, the $D_F \times D_F \times M$ input feature map is convolved with N independent $D_K \times D_K \times M$ convolution filters. Each filter produces a $D_G \times D_G \times 1$ output feature map, the output feature map of this layer is the stack of N output feature maps and it has a dimension of $D_G \times D_G \times N$, where D_G is given by:

$$D_G = \frac{D_F - D_K + 2P}{S} + 1 \quad (2.14)$$

where S denotes the stride of the convolution filters. P denotes the zero padding size (i.e., zero padding means padding the input feature map with zeros around the border). The depth of the output feature map is determined by the number of convolution filters applied in the convolutional layer. Taking the first convolutional layer shown in Figure 2.2 as an example, the input feature map of the first convolutional layer is a $227 \times 227 \times 3$ image. The first convolutional layer contains 96 filters with a stride of 4 and the dimension of each filter is $11 \times 11 \times 3$. The spatial width or height of the output feature map of this layer is thus calculated by $D_G = \frac{227-11+2 \times 0}{4} + 1 = 55$

(Assuming no zero padding is applied in this layer). The depth of the output feature map is 96 since there are 96 convolution filters in this layer. The size of the output feature map of this layer is thus $55 \times 55 \times 96$, as illustrated in Figure 2.2.

To measure the computational cost of a layer, a widely used metric is the number of multiply-adds [3, 4, 50, 8]. In this thesis, we use MAdds to denote the number of multiply-adds for simplicity. The MAdds of a convolutional layer is given in [3], as shown below:

$$\text{MAdds} = C_{in} \times K^2 \times H_{out} \times W_{out} \times C_{out} \quad (2.15)$$

where K is the width or height of the convolution filters, C_{in} is the depth of the input feature map, H_{out} , W_{out} and C_{out} are the height, width and depth of the output feature map respectively.

The MAdds of the typical convolutional layer shown in Figure 2.4 is calculated using Eq. (2.15), as shown below:

$$\text{MAdds} = M \times D_K^2 \times D_G^2 \times N \quad (2.16)$$

The deployment of deep learning models in real-world applications is not only constrained by MAdds. Another important factor is the number of learnable parameters [3, 50, 8, 51]. In this thesis, we use Parameters to denote the number of learnable parameters for simplicity. In general, the strong representation ability of a CNN model comes from the millions of learnable parameters. The learnable parameters and the model structure need to be stored on the disk and loaded into memory during the training and testing process. Storing a typical CNN (ResNet [2], GoogleNet [34], VGGNet [52], etc) needs more than 300MB space. This may not be a problem for the high-end GPU (graphics processing unit), but it is unaffordable for the resource-

constrained devices such as mobile devices or Internet of Things (IoT) devices [53]. Given equivalent performance, a CNN architecture with fewer learnable parameters has three main advantages [51]:

- Less communication overhead for distributed training.
- Fewer data transfers when exporting updated models to clients.
- Feasible resource-constrained devices deployment.

The overall Parameters for the standard convolutional layer shown in Figure 2.4 is given by:

$$\text{Parameters} = D_K^2 \times M \times N \quad (2.17)$$

2.2.2 Depthwise Separable Convolutional Layer

The standard convolutional layers are widely applied in many modern CNN architectures (e.g., AlexNet, ResNet [2]) and the CNN models are becoming deeper in order to obtain better performance [2, 34]. On the other hand, the MAdds or Parameters of a very deep CNN (a CNN model with many standard convolutional layers) may become unaffordable. In [54], a computationally efficient convolutional layer, namely the depthwise separable convolutional layer, was proposed to tackle this problem. It has been shown that by replacing standard convolutional layers with depthwise convolutional layers in AlexNet, the accuracy remains approximately the same while the training speed is significantly faster [54]. The difference between a standard convolutional layer and a depthwise convolutional layer is described below.

For the standard convolutional layer, the convolution filters across the entire depth of the input feature map and the weighted sum are carried out in a single step. On

Figure 2.3: Illustration of convolving a single $7 \times 7 \times 3$ filter over a $32 \times 32 \times 3$ input feature map with a stride of 1 (shifting the filter one unit horizontally or vertically at a time). There are 26×26 spatial locations for a $7 \times 7 \times 3$ filter to slide over a $32 \times 32 \times 3$ input feature map, thus this procedure generates a $26 \times 26 \times 1$ output feature map, where each element in the output feature map is the sum of the element-wise multiplication of the filter and the small patch of the input feature map it overlapped.

the other hand, the depthwise separable convolutional layer decomposes the standard convolution into a depthwise convolution step and a pointwise convolution step, as described below:

1. **Depthwise Convolution Step:** In the depthwise convolution step, the input feature map is convolved with a depthwise convolutional layer, as illustrated in Figure 2.5. Suppose the input feature map has a dimension of $D_F \times D_F \times M$. The depthwise convolutional layer regards the $D_F \times D_F \times M$ input feature map as M independent $D_F \times D_F \times 1$ input feature maps. Each $D_F \times D_F \times 1$ input feature map is then convolved with a $D_K \times D_K \times 1$ convolution filter to create a $D_G \times D_G \times 1$ output feature map. This process will generate M independent $D_G \times D_G \times 1$ output feature maps. We stack those output

Figure 2.4: Illustration of convolving a $D_F \times D_F \times M$ input feature map with a standard convolutional layer. This convolutional layer contains N convolution filters. Each filter has a dimension of $D_K \times D_K \times M$.

Figure 2.5: Illustration of convolving a $D_F \times D_F \times M$ input feature map with a depthwise convolutional layer. The depthwise convolutional layer contains M convolutional filters. Each filter has a dimension of $D_K \times D_K \times 1$.

feature maps to form the output feature map of this layer. Thus the output feature map of this depthwise convolutional layer has a dimension of $D_G \times D_G \times M$. The MAdds of this depthwise convolution layer is $D_K^2 \times D_G^2 \times M$. The Parameters of this layer is $D_K \times D_K \times M$.

2. **Pointwise Convolution Step:** In the pointwise convolution step, the output feature map generated by the depthwise convolution step is convolved with a pointwise convolutional layer, as illustrated in Figure 2.6. In this step, the output feature map generated by the depthwise convolution layer is convolved with $1 \times 1 \times M$ convolution filters. The $1 \times 1 \times M$ convolution filter is called pointwise convolution filter. Each pointwise convolution filter generates a $D_G \times D_G \times 1$ output feature map. We apply N such pointwise filters to generate N output feature maps and those output feature maps are stacked to form the output feature map of this pointwise convolutional layer. Thus the output feature map of this pointwise convolutional layer has a dimension of $D_G \times D_G \times N$. The MAdds of this layer is $D_G^2 \times M \times N$. The Parameters of this layer is $M \times N$.

The total MAdds of the depthwise separable convolutional layer is the sum of the MAdds of the depthwise convolutional layer and the MAdds of the pointwise convolutional layer. The comparison of MAdds of the standard convolutional layer illustrated in Figure 2.4 and MAdds of the depthwise separable convolutional layer illustrated in Figure 2.5 and Figure 2.6 is summarized below:

Figure 2.6: Illustration of convolving a $D_G \times D_G \times M$ input feature map with a pointwise convolutional layer. The pointwise convolutional layer contains N convolutional filters. Each filter has a dimension of $1 \times 1 \times M$.

MAdds of the standard convolutional layer : $D_K^2 \times D_G^2 \times M \times N$.

MAdds of the depthwise separable convolutional layer : $D_K^2 \times D_G^2 \times M + D_G^2 \times M \times N$.

$$\frac{\text{MAdds of the depthwise separable convolutional layer}}{\text{MAdds of the standard convolutional layer}} = \frac{1}{N} + \frac{1}{D_K^2}.$$

The Parameters of the depthwise separable convolutional layer is the sum of the Parameters of the depthwise convolutional layer and the Parameters of the pointwise convolutional layer. The comparison of the Parameters of the standard convolutional layer and the Parameters of the depthwise separable convolutional layer is summarized below:

Parameters of the standard convolutional layer : $D_K^2 \times M \times N$.

Parameters of the depthwise separable convolutional layer : $M \times D_K^2 + M \times N$.

$$\frac{\text{Parameters of the depthwise separable convolutional layer}}{\text{Parameters of the standard convolutional later}} = \frac{1}{N} + \frac{1}{D_K^2}.$$

It can be seen that the MAdds or Parameters of the depthwise separable convolutional layer is $\frac{1}{N} + \frac{1}{D_K^2}$ of those of the standard convolutional layer [3]. For modern CNNs, the depth of the output feature map N is normally much greater than D_K^2 (Typically $N \geq 32$ and $D_K = 3$). Thus, the MAdds or Parameters of the depthwise separable convolutional layer is approximately $\frac{1}{D_K^2} = \frac{1}{9}$ of those of the standard convolutional layer.

2.2.3 Max Pooling Layer

Max pooling layer is another core layer of CNNs and it aims to reduce the width and height of the input feature map by max pooling operations (i.e., taking the max value from sub-regions of the input feature map). It also reduces the over-fitting problem

by providing a down-sampled feature map. Max pooling works by sliding a window across the input feature map and output the maximum value of the overlapped sub-regions. Figure 2.7 illustrates the max pooling operations with a $2 \times 2 \times 1$ pooling filter and a stride of 2 over an $4 \times 4 \times 1$ input feature map.

There are three max pooling layers applied in the AlexNet architecture, as shown in Figure 2.2. Considering the first max pooling layer as an example, 3×3 max pooling filters with stride of 2 are applied to the $55 \times 55 \times 96$ feature map. The resulting output feature map of this pooling layer is a $27 \times 27 \times 96$ feature map. Note that pooling layer does not change the depth of the feature map, it only downsamples the width and height of the input feature map.

Figure 2.7: Illustration of maxpooling operations.

2.2.4 Fully Connected Layer

The fully-connected layer provides a non-linear combination of the extracted features so that the output layer can use these features to classify the input vector into the corresponding class. The fully connected layers are usually placed after the last convolutional layer or pooling layer as shown in Figure 2.2. The fully connected

layers are the same as the fully connected layers in the typical FCNN discussed in Section 2.1.

The MAdds of a fully connected layer is given by:

$$\text{MAdds} = I \times O \quad (2.18)$$

The number of parameters of a fully connected layer is given by:

$$\text{Parameters} = (I + 1) \times O \quad (2.19)$$

where I is the input dimensionality and O is the output dimensionality in Eq.(2.18) and Eq.(2.19).

2.3 Typical Workflow of a Deep Learning Project

The basic concepts of neural networks are reviewed in the previous section. In this section, we summarize the typical workflow of applying neural network models in a real-world application. The typical workflow is summarized as follows:

1. **Data Acquisition:** First, prepare a dataset for the application. The dataset is then split into three parts, namely the training set, the validation set and the test set. The training set is used for training the model with backpropagation algorithms. The validation set is used for hyperparameter optimization and the test set is used for performance evaluation.
2. **Exploratory Data Analysis:** After we obtained the dataset, exploratory data analysis (EDA) is performed in order to discover latent patterns such as imbalanced data distribution, features at different scales, identifying anomalies or outliers using statistical methods and illustrate data properties using graphical representations.

3. **Data Preprocessing and Data Augmentation:** After EDA, data preprocessing techniques and data augmentation techniques are applied on the dataset. Preprocessing techniques are widely used for improving the convergence of the neural network model [55]. For image dataset, standardization (subtracting the mean and dividing by the standard deviation individually for RGB channel) is commonly used. Data augmentation techniques are used to generate artificial training samples and expand the training dataset. It has been widely used for improving the model performance and reducing overfitting, especially for training with a small dataset or imbalanced dataset [56]. Commonly used image augmentation techniques include image cropping, image flipping, image rotating, etc.
4. **Architecture Design:** Next, we need to find a set of architectures which are suitable for the applications. If we have limited computational resources or a real-time prediction is needed, we may use computationally efficient CNN models for the task. On the other hand, we may use very deep CNN models to obtain good performance if adequate computational resources are provided.
5. **Hyperparameter Optimization and Training:** Next, we need to choose a set of hyperparameters to train the model. Hyperparameters are set of non-learnable parameters which need to be predefined before the training process begins. The hyperparameter optimization process consists of sampling hyperparameters from the manually specified subset of the hyperparameter space, training the models with different hyperparameter settings and evaluating those models on the validation set. The combination of the hyperparameters which yields an optimal model is selected as the optimal hyperparameter set. The final model is the model trained with the optimal hyperparameter set.

6. **Performance Evaluation:** After the model is trained using optimal hyperparameters, the performance of the final model is evaluated using the unseen test set.
7. **Ensemble Learning (optional):** Consistent performance improvement can be obtained by applying ensemble learning (i.e., ensemble learning refers to use multiple models to obtain better performance than any single model [57]). Commonly used ensemble learning techniques include Bagging (i.e., voting with equal weight), Stacking (i.e., training a meta-classifier to combine the predictions of several base-level models), etc.

Chapter 3

MobileNet-Dense: An Efficient Convolutional Neural Network

In this chapter, we propose and assess the performance of a novel CNN architecture MobileNet-Dense. In Section 3.1, we review the related work. In Section 3.2, we introduce the MobileNet-Dense model based on dense connectivity and depthwise separable convolutional layers. The effectiveness of MobileNet-Dense model is illustrated in Section 3.3 using two benchmark datasets.

3.1 Related Works

We first review the CNNs with skip connections (i.e., ResNet and DenseNet). Next, we review the computationally efficient CNN model constructed using depthwise separable convolutional layers (i.e., MobileNetV1) and the computationally efficient CNN model constructed using both depthwise separable convolutional layers and residual skip connections (i.e., MobileNetV2).

3.1.1 CNN with Skip Connections: ResNet and DenseNet

As computer vision tasks become increasingly complex, state-of-the-art CNNs are becoming rapidly deeper. The VGGNet [52] extends the depth of AlexNet from eight to nineteen layers and achieves an error rate of 7.3% compared to an error rate of 16.4% achieved by AlexNet on ImageNet dataset. GoogleNet [34] achieves an error rate of 6.7% on ImageNet dataset and it extends the depth in a different

Table 3.1: Error rates of plain CNNs and ResNets on ImageNet dataset [2]

	Plain CNN	ResNet
18 layers	27.94%	27.88%
34 layers	28.54%	25.03%

way. GoogleNet concatenates the output feature map produced by filters of different sizes to enrich the features learned from the input feature map at each layer, but the general idea is similar, namely to boost the performance by extending the depth of the CNNs. Although making CNNs deeper generally improves the performance, this is not always true. Simply adding more layers to an appropriately deep CNN may result in higher training and testing errors [58, 59, 2].

ResNet

In order to tackle this performance degradation problem, a Residual Network (i.e., CNN with residual skip connections) was proposed in [2]. In [2], a 18-layer plain CNN (i.e., CNN without residual skip connections), a 34-layer plain CNN, a 18-layer ResNet and a 34-layer ResNet were trained and compared on ImageNet dataset. The results are shown in Table 3.1. It can be seen that adding 16 layers to a 18-layer ResNet leads to improved performance while adding 16 layers to a 18-layer plain CNN leads to degraded performance. In addition, both 18-layer ResNet and 34-layer ResNet outperform their plain counterparts. The results indicate that residual connections are effective not only for avoiding the performance degradation problem but also for boosting the performance.

The basic bottleneck blocks for the ResNet architecture are shown in Figure 3.1, in which the skip connections are those connections which skip one or more layers. In the rest of this thesis, we use ConvNxN layer to denote a convolutional layer which contains multiple $N \times N$ convolution filters. The basic building block of

ResNet model is called the bottleneck block since Conv1x1 layer is applied as a 'bottleneck' to reduce the depth of the input feature map and thereby improving the computational efficiency. We first explain the residual bottleneck layer (stride=1) shown in Figure 3.1. Suppose the input for this bottleneck layer is a $D_F \times D_F \times M$ feature map and it is denoted by $\mathbf{F}_{in}$. It can be seen that there are two paths for $\mathbf{F}_{in}$ to go forward. The main path (illustrated by blue arrows) is formed by a stack of standard convolutional layers and the skip connection path (illustrated by green arrows) is simply performing an identity mapping. In the main path, the first Conv1x1 layer is used for reducing the depth of the input feature map proportionally by a factor of t ($t < 1$) and thus it generates a $D_F \times D_F \times tM$ feature map $\mathbf{F}_1$. The following Conv3x3 layer generates a $D_F \times D_F \times tM$ feature map $\mathbf{F}_2$. The last Conv1x1 layer is used for expanding (restoring) the depth of the feature map and it generates a $D_F \times D_F \times M$ feature map $\mathbf{F}_3$. The output feature map $\mathbf{F}_{out}$ is obtained by performing element-wise addition of $\mathbf{F}_{in}$ and $\mathbf{F}_3$. For residual bottleneck layer (stride=2), a Conv3x3 layer with a stride of 2 is applied in the main path to downsample the width and height of the feature map. Therefore, a Conv1x1 layer with a stride of 2 is applied in the skip connection path to match the dimensions.

The ResNet involves skip connections to let intermediate convolutional layers learn a residual mapping $f(x) = H(x) - x$ rather than a desired underlying mapping as $H(x)$, which solves the performance degradation problem [2]. To explain why, suppose a ResNet with l residual bottleneck blocks is sufficient for a specific task and we construct a ResNet with $l + m$ residual bottleneck blocks, then ideally the backpropagation algorithm will force the residual to zero for all the m redundant bottleneck blocks and thus the last m redundant bottleneck blocks are simply performing identity mapping. Therefore the performance of the ResNet with $l + m$

Figure 3.1: Building blocks of ResNet [2].

residual bottleneck blocks should not worse than the performance of the ResNet with l residual bottleneck blocks.

DenseNet

A variant of residual skip connection, namely the dense connection, is proposed in the DenseNet architecture [8]. The two types of building blocks of the DenseNet architecture, namely the bottleneck block and the transition block, are shown in

Figure 3.2: Building blocks of DenseNet [8].

Figure 3.2. For bottleneck block, the first Conv1x1 layer in the main path is convolved with a $D_F \times D_F \times M$ input feature map $\mathbf{F}_{in}$ and it generate a $D_F \times D_F \times 4K$ feature map $\mathbf{F}_1$. The following Conv3x3 layer generates a $D_F \times D_F \times K$ feature map $\mathbf{F}_2$. The input feature map $\mathbf{F}_{in}$ is then concatenated with the feature map $\mathbf{F}_2$ along the depth axis to form the output feature map $\mathbf{F}_{out}$. Thus $\mathbf{F}_{out}$ is a $D_F \times D_F \times (M + K)$ feature map. A dense block is constructed using multiple bottleneck blocks, as shown in Figure 3.3. It can be seen that all bottleneck blocks within a

dense block are directly connected with each other. In other words, each bottleneck block receives the output feature map from all preceding bottleneck blocks and passes on its output feature map to all subsequent bottleneck blocks. This connectivity is called the dense connectivity. Compared to the traditional connectivity (i.e., each building block only receives output feature map from the nearest preceding block and passes on its output feature map to the nearest subsequent block), the dense connectivity allows more feature reuse and promote information flow throughout the network by introducing more connections [8]. A DenseNet model is constructed using a stack of these dense blocks, as shown in Figure 3.4. Between two adjacent dense blocks, a transition block is applied to downsample the size of the feature map. It is found that DenseNet achieves an Acc of 77.5% compared to an Acc of 78.2% achieved by ResNet on ImageNet dataset while using 55% fewer parameters and 49% fewer MAdds [60].

3.1.2 Efficient CNNs: MobileNetV1 and MobileNetV2

Despite the great success of modern CNNs (e.g. ResNet, DenseNet, etc.) in many image recognition applications, one noteworthy drawback of such CNNs is the high computational cost. In this section, we review the computationally efficient MobileNetV1 [3] model which is constructed using depthwise separable convolutional layers and MobileNetV2 model constructed using depthwise separable convolutional layers and residual skip connections.

MobileNetV1

The structure of MobileNetV1 [3] is described in Table 3.2 and the basic building block is shown in Figure 3.5. The MobileNetV1 model is basically a CNN model with a stack of depthwise separable convolutional layers. A hyperparameter α is

Figure 3.3: Illustration of a typical Dense Block [8] constructed using 5 bottleneck blocks. Each bottleneck block is illustrated by a single blue node. Any two of the nodes within the same dense block are directly connected.

introduced to reduce($\alpha < 1$) or increase($\alpha > 1$) the model complexity by altering the depth of the feature map for each building block proportionally by a factor α . The α is applied to trade off model performance and computational cost. Unlike the traditional CNN architecture, the MobileNetV1 architecture does not apply any pooling layers since it has been shown in [61] that pooling layers can be replaced by convolutional layers with an increased stride without loss in accuracy. Therefore, a depthwise Conv3x3 layer with a stride of 2 is applied to downsample the spatial width and height of the feature map.

Figure 3.4: A deep DenseNet [8] with two dense blocks. The Conv1x1 layer and pooling layer between two dense blocks are applied to downsample the size of the feature map.

MobileNetV2

Sandler et al. [4] proposed the MobileNetV2 architecture constructed using depthwise separable convolutional layers with residual skip connections. The two types of building blocks for MobileNetV2, namely the inverted residual building block with linear bottlenecks and the reduction building block with linear bottlenecks, are illustrated in Figure 3.6. We briefly review these two building blocks below:

- The Inverted Residual Block with linear bottleneck:** The inverted residual block begins with a Conv1x1 layer which is used for expanding the depth of the input feature map by a factor of t ($t > 1$). Then a depthwise Conv3x3 layer is convolved with the expanded feature map. The last Conv1x1 layer is used for compressing (restoring) the depth of the feature map to allow for the element-wise addition. If the expansion ratio t is set to be less than 1, this is the classical residual bottleneck block (shown in Figure 3.1). The main reason for expanding the depth of the feature map at first Conv1x1 layer is to prevent the information loss caused by the presence of ReLU activation functions [4]. It has been shown that expanding the depth of the feature map with a sufficiently large expansion rate and applying Relu on the expanded feature map is resistant to the information loss [4]. In addition, the inverted residual building block is more memory efficient than the classical residual block [4]. The

Figure 3.5: Building block of MobileNetV1 [3].

Table 3.2: The structure of MobileNetV1 [3]

Input	Operator	M	r	s
$672^2 \times 3$	Conv3x3 Layer	32	1	2
$336^2 \times 32$	Building Block	64	1	1
$336^2 \times 64$	Building Block	128	1	2
$168^2 \times 128$	Building Block	128	1	1
$168^2 \times 128$	Building Block	256	1	2
$84^2 \times 256$	Building Block	256	1	1
$84^2 \times 256$	Building Block	512	1	2
$42^2 \times 512$	Building Block	512	5	1
$42^2 \times 512$	Building Block	1024	1	2
$21^2 \times 1024$	GlobalAvgPool Layer	1024	1	
$1^2 \times 1024$	Output Layer	-	-	

Suppose the input is a $672^2 \times 3$ image. Each line in this table describes a layer or a sequence of building block repeated r times. M denotes the depth of the output feature map for each layer or sequence. s denotes the stride of the Conv3x3 layer.

linear bottleneck refers to replacing the Relu non-linear activation with linear activation on the output feature map of the last Conv1x1 layer. The experimental result showed that applying linear activation on the output feature map of the last Conv1x1 layer lead to improved performance since it also prevents the information loss caused by Relu non-linear activation [4].

- **The Reduction Block with linear bottleneck:** The reduction block of MobileNetV2 begins with a Conv1x1 layer which is used for expanding the depth of the input feature map by a factor t ($t > 1$), followed by a depthwise Conv3x3 layer a Conv1x1 layer with linear activation. The depthwise Conv3x3 layer is responsible for downsampling the width and height of the feature map and the last Conv1x1 layer is responsible for compressing the depth of the feature map. The reduction block does not include a residual skip connection since the spatial width and height of its output feature map $\mathbf{F}_{out}$ are smaller than those of the input feature map $\mathbf{F}_{in}$.

The standard MobileNetV2 architecture is described in Table 3.3. It can be seen that the MobileNetV2 model is constructed using a stack of residual modules. Each residual module is constructed using multiple building blocks, where the first building block can be either an inverted residual block or a reduction block and the following building blocks are inverted residual blocks. Similar to MobileNetV1 architecture, a hyperparameter α is introduced to trade off performance and computational cost. It is found that MobileNetV2 achieves an Acc of 0.720 compared to an Acc of 0.706 achieved by MobileNetV1 while using 19% fewer parameters and 48% fewer MAdds on ImageNet dataset [4].

Figure 3.6: Building blocks of MobileNetV2 [4].

Table 3.3: The structure of MobileNetV2 [4]

Input	Operator	M	r	s	t
$672^2 \times 3$	Conv3x3 Layer	32	1	2	-
$336^2 \times 32$	Residual Module	16	1	1	1
$336^2 \times 16$	Residual Module	24	2	2	6
$168^2 \times 24$	Residual Module	32	3	2	6
$84^2 \times 32$	Residual Module	64	4	2	6
$42^2 \times 64$	Residual Module	96	3	1	6
$42^2 \times 96$	Residual Module	160	3	2	6
$21^2 \times 160$	Residual Module	320	1	1	6
$21^2 \times 320$	Conv1x1 Layer	1280	1	1	-
$21^2 \times 1280$	GlobalAvgPool Layer	1280	1	-	-
$1^2 \times 1280$	Output Layer	-	-	-	-

Suppose the input is a $672^2 \times 3$ image. Each line in this table describes a layer or a residual module. Each residual module is constructed using r building blocks, where the first building block can be either an inverted residual block or a reduction block and the following $(r-1)$ building blocks are inverted residual blocks. M denotes the depth of the output feature map for each layer or sequence. s denotes the stride of the Conv3x3 layer. t denotes the expansion rate of each building block. Except for the first residual module, a constant expansion rate is applied throughout the network.

3.2 MobileNet-Dense: An efficient CNN based on Dense Connections and Depthwise Separable Convolutional Layers

As discussed in Section 3.1, DenseNet achieves a similar performance but with fewer MAdds and Parameters as ResNet on ImageNet dataset. Also, MobileNetV2 achieves better results than MobileNetV1 through the addition of residual connections [4]. Based on these two observations, we propose a novel MobileNet-Dense model which is constructed by adding dense connectivity to the inverted bottleneck block.

The core building blocks of the proposed MobileNet-Dense, namely the bottleneck block and the reduction block, are shown in Figure 3.7. These two building blocks are discussed as follows:

- **The Bottleneck Block:** In order to prevent the information loss caused by Relu activation, the design of the bottleneck block follows the design of inverted residual block [4]. Specifically, the bottleneck block of MobileNet-Dense is constructed using a stack of 3 convolutional layers. The three convolutional layers are Conv1x1 layer, depthwise Conv3x3 layer, and Conv1x1 layer with linear activation respectively. The first Conv1x1 layer is responsible for expanding the depth of the feature map by a factor of t so as to prevent the information loss. The last Conv1x1 layer with linear activation is responsible for compressing the depth of the feature map so as to improve the computation efficiency. Following the design of DenseNet, we let each bottleneck block generates a feature map with K channels. In other words, the last Conv1x1 layer compressed the depth of the feature map to be a fixed value K . Then the input feature map $\mathbf{F}_{in}$ and the output feature map of the last Conv1x1 layer $\mathbf{F}_3$ are concatenated along the depth axis to allow for the dense connectivity. The hyperparameter K can

Figure 3.7: Building blocks of MobileNet-Dense

be used to trade off the performance and the computational cost.

- **The Reduction Block:** The Reduction Block begins with a Conv1x1 layer, followed by a depthwise Conv3x3 layer with a stride of 2 and a Conv1x1 layer with linear activation. The first and the last Conv1x1 layers are used for expanding and then compressing the depth of the feature map. The depthwise Conv3x3 with a stride of 2 is used for downsampling the width and height of the feature map. The reduction block does not include the skip connection (i.e., feature concatenation) since the spatial width and height of its output feature map F_{out} are smaller than those of the input feature map F_{in} .

The MobileNet-Dense is constructed using a stack of dense modules, where each dense module is constructed by densely connecting r building blocks. To be more specific, each dense module begins with one reduction block to downsample the width and height of the feature map, followed by $(r - 1)$ bottleneck blocks. Between two adjacent dense modules, a Conv1x1 layer with linear activation is applied to halve the depth of the feature map in order to improve the computational efficiency. The structure of the proposed MobileNet-Dense is summarized in Table 3.4.

3.3 Experiment

We empirically demonstrate the effectiveness of MobileNet-Dense on two widely used benchmark datasets, namely CIFAR-10 and CIFAR-100. We select the MobileNetV2 model as the benchmark model to compare the results.

3.3.1 Experiment Environment

The hardware environment and software environment used in our experiment are summarized below:

Table 3.4: The structure of MobileNet-Dense

Input	Operator	M	r	s	K	t
$672^2 \times 3$	Conv3x3 Layer	32	1	2	-	-
$336^2 \times 32$	Bottleneck Block	48	1	1	16	1
$336^2 \times 48$	Dense Module	96	2	2	48	3
$168^2 \times 96$	Conv1x1 Layer	48	1	1	-	-
$168^2 \times 48$	Dense Module	144	3	2	48	3
$84^2 \times 144$	Conv1x1 Layer	72	1	1	-	-
$84^2 \times 72$	Dense Module	216	4	2	48	3
$42^2 \times 216$	Conv1x1 Layer	108	1	1	-	-
$42^2 \times 108$	Dense Module	300	5	2	48	3
$21^2 \times 300$	Conv1x1 Layer	1280	1	1	-	-
$1^2 \times 1280$	GlobalAvgPool Layer	1280	1	-	-	-
$1^2 \times 1280$	Output Layer	-	-	-	-	-

Suppose the input is a $672^2 \times 3$ image. Each line in this table describes a layer or a dense module. Each dense module is constructed using r building blocks, where the first building block is a reduction block and the following $(r - 1)$ building blocks are bottleneck blocks. We constructed the standard MobileNet-Dense model using 4 dense modules, where those dense modules contain 2, 3, 4 and 5 building blocks respectively. M denotes the depth of the output feature map for each layer or dense module. K denotes the number of convolution filters applied in the last Conv1x1 layer of each bottleneck block. t denotes the expansion rate of each building block. The K and t of the first bottleneck block (the 2nd line of this table) are fixed to 16 and 1 respectively, the K and t of the following 4 dense modules need to be grid searched (In this table we set $K = 48$ and $t = 3$ for illustration purposes).

1. **Hardware:**

- CPU (Central Processing Unit): Intel i7-6850K Processor
- RAM (Random-access Memory): Kingston 48GB DDR4
- GPU (Graphics Processing Unit): Nvidia GeForce GTX 1080 Ti

2. **Software:**

- OS (Operating System): Ubuntu 16.04.4 LTS
- Coding Environment: Python 2.7
- Core Library: CUDA 9.0, Cudnn 7.0, Tensorflow-gpu 1.9.0, Keras 2.2.0

3.3.2 CIFAR Datasets

The CIFAR-10 dataset and CIFAR-100 dataset consist of colored RGB natural images, each image has a size of 32×32 pixels. CIFAR-10 consists of 10 classes, each with 6000 images. CIFAR-100 consists of 100 classes, each with 600 images. For both CIFAR-10 and CIFAR-100, the training set contains 50000 images and the test set contains 10000 images.

3.3.3 Training

Following the typical workflow described in Section 2.3, we first apply preprocessing and augmentation techniques on CIFAR datasets. For preprocessing, we standardized the image data using the RGB channel means and standard deviations. For data augmentation, we used two image augmentation techniques (mirroring/shifting) which have been widely used on these two datasets [2, 8, 62]. As the size of the images in CIFAR-10 and CIFAR-100 datasets is only 32×32 pixels, we changed the stride

of the first Conv3x3 layer from 2 to 1 for both MobileNetV2 model and MobileNet-Dense model and we removed the first bottleneck block with a stride of 2 for both MobileNetV2 model and MobileNet-Dense model to preserve the spatial width and height of the feature map at the beginning of the network. The MobileNetV2 model and MobileNet-Dense model we trained for CIFAR datasets are summarized in Table 3.5 and Table 3.6 respectively. There are two hyperparameters (i.e., t and α) for MobileNetV2 and there are two hyperparameters (i.e., t and K) for MobileNet-Dense. Grid search is applied to optimize the hyperparameter values. Considering the computational cost, we defined the searching space as below:

$$t \in \{2, 3, 4, 5, 6\}$$

$$K \in \{16, 32, 48, 64\}$$

$$a \in \{1.0, 1.1, 1.2, 1.3\}$$

All the networks with different hyperparameter values are trained using the Adam optimizer [63]. We trained each network using a batch size of 64 for 200 epochs. The initial learning rate is set to 0.001 as suggested in [63], and it is successively decreased by 10 at 40%, 60% and 80% of the total number of training epochs. The hyperparameters set we found for MobileNet-Dense is $[t = 3, K = 48]$ and the hyperparameters set we found for MobileNetV2 is $[t = 6, \alpha = 1.3]$.

3.3.4 Performance Evaluation

The performance of MobileNet-Dense and MobileNetV2 on the CIFAR-10 and CIFAR-100 datasets are summarized in Table 3.7.

As we can see that the MobileNet-Dense and MobileNetV2 achieve similar accuracies on both CIFAR-10 and CIFAR-100 datasets, while MobileNet-Dense using

Table 3.5: The structure of MobileNetV2 for CIFAR datasets

Input	Operator	M	r	s	t
$32^2 \times 3$	Conv3x3 Layer	40	1	1	-
$32^2 \times 40$	Residual Module	24	1	1	1
$32^2 \times 24$	Residual Module	32	1	1	6
$32^2 \times 32$	Residual Module	40	3	2	6
$16^2 \times 40$	Residual Module	80	4	2	6
$8^2 \times 80$	Residual Module	128	3	1	6
$8^2 \times 128$	Residual Module	208	3	2	6
$4^2 \times 208$	Residual Module	416	1	1	6
$4^2 \times 416$	Conv1x1 Layer	1664	1	1	-
$4^2 \times 1664$	GlobalAvgPool Layer	1664	1	-	-
$1^2 \times 1664$	Softmax Layer	-	-	-	-

The input is a $32^2 \times 3$ image. The differences between MobileNetV2 for CIFAR and standard MobileNetV2 model described in Table 3.3 are highlighted in red color.

Table 3.6: The structure of MobileNet-Dense for CIFAR datasets

Input	Operator	M	r	s	K	t
$32^2 \times 3$	Conv3x3 Layer	32	1	1	-	-
$32^2 \times 32$	Bottleneck Block	48	1	1	16	1
$32^2 \times 48$	Bottleneck Block	96	1	1	48	3
$32^2 \times 96$	Conv1x1 Layer	48	1	1	-	-
$32^2 \times 48$	Dense Module	144	3	2	48	3
$16^2 \times 144$	Conv1x1 Layer	72	1	1	-	-
$16^2 \times 72$	Dense Module	216	4	2	48	3
$8^2 \times 216$	Conv1x1 Layer	108	1	1	-	-
$8^2 \times 108$	Dense Module	300	5	2	48	3
$4^2 \times 300$	Conv1x1 Layer	1280	1	1	-	-
$4^2 \times 1280$	GlobalAvgPool Layer	1280	1	-	-	-
$1^2 \times 1280$	Softmax Layer	-	-	-	-	-

The input is a $32^2 \times 3$ image. The differences between MobileNet-Dense for CIFAR and standard MobileNet-Dense model described in Table 3.4 are highlighted in red color.

Table 3.7: Performance on CIFAR-10 and CIFAR-100 datasets

Method	MAdds ¹	Parameters ²	Acc (CIFAR-10)	Acc (CIFAR-100)	Testing Time ³
MobileNetV2	138M	3.7M	0.931	0.731	325 μs
MobileNet-Dense	89M	1.4M	0.934	0.727	270 μs

¹ MAdds is in Millions and it is the MAdds for predicting one image.

² Parameters is in Millions.

³ Testing time is in microseconds and it is the inference time for predicting one image.

62% fewer learnable parameters and 36% fewer MAdds compared to MobileNetV2. In terms of testing time per image, the MobileNet-Dense is 17% faster than the MobileNetV2. The results indicate that the proposed MobileNet-Dense is effective and efficient.

3.4 Summary

In this Chapter, a novel CNN model, namely MobileNet-Dense was proposed. The classification performance and the computational cost of MobileNet-Dense are compared with those of the MobileNetV2 model using the CIFAR-10 and the CIFAR-100 datasets. It is found that MobileNet-Dense and MobileNetV2 model achieves similar classification performance on those two benchmark datasets, while MobileNet-Dense model using 62% fewer parameters, 36% fewer MAdds and 17% lesser running time compared to MobileNetV2 model. These results indicate that the proposed MobileNet-Dense model is effective and efficient.

Chapter 4

Automatic Diabetic Retinopathy Classification System Design

This chapter describes the design of the computationally efficient DR classification system. The workflow of the DR classification system, as shown in Figure 4.1, follows the typical workflow discussed in Section 2.3. The structure of this chapter follows the workflow illustrated in Figure 4.1. Section 4.1 describes the exploratory data analysis (EDA) of the EyePACS dataset [5]. In Section 4.2, data augmentation and image preprocessing techniques used in our system are described. In Section 4.3, the hyperparameter tuning and the training details are discussed. In Section 4.4, the ensemble learning method employed in our system is described. A summary is provided in Section 4.5.

4.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) [64] is a core process for general machine learning tasks. It refers to the primary inspection process of the data in order to discover underlying patterns such as imbalanced data distribution, identify anomalies or outliers using statistical methods and illustrates data properties using graphical representations. As the EyePACS DR dataset [5] contains retinal images and the corresponding DR severity labels, we perform basic EDA on both the retinal images and the labels.

Figure 4.1: Flowchart of the proposed automatic DR classification system.

Table 4.1: DR label information in EyePACS dataset [5]

DR Severity Level	DR Severity Label
No DR	0
Mild DR	1
Moderate DR	2
Severe DR	3
Proliferative DR	4

4.1.1 EDA of Training Labels

The EyePACS DR dataset [5] is a large DR dataset with high-resolution images taken under different imaging conditions. There are 17653/5453/21335 pairs of color images for the training/validation/test set and the corresponding DR severity levels are provided. As shown in Table 4.1, the DR levels are rated from 0 to 4 based on the severity of the DR disease assessed by an ophthalmologist. The distribution of DR levels in the training dataset is illustrated in Figure 4.2. It can be seen that the distribution is very non-uniform, with 73% of the images labeled as healthy, 6% as mild DR, 15% as moderate DR, 2% as severe DR and 2% as proliferative DR. The EyePACS DR training set also shows that the DR labels for a given pair of eyes are highly correlated, as shown in Figure 4.3. To be more specific, it can be seen that 87% of pairs of eyes have the same DR severity label, 95% of pairs of eyes have the DR label values differ by at most 1.

4.1.2 EDA of Training Images

Random sample retinal images with different DR levels from the EyePACS DR training set are shown in Figure 4.4. Each row contains five retinal images with the same DR level. Retinal images in this dataset were taken in different illumination conditions or by different types of cameras [5]. Therefore the size of the image and the light conditions vary (e.g., the 4th image with moderate DR, the 2nd and the 3rd

Figure 4.2: Distribution of DR label values in EyePACS training set.

image with proliferate DR in Figure 4.4 are relatively dark compared with the other images). The sizes of the training images varies from 400×315 to 5184×3456 . For illustration purposes, all images shown in Figure 4.4 are resized to 672×672 using bilinear interpolation. The dataset contains some underexposed images (i.e., images which are too dark) and overexposed images (i.e., images which are too bright) [5]. It is useful to check whether there are many images which are poorly exposed since existing deep learning models are shown to be sensitive to image quality [65].

We converted the RGB images to 8-bit grayscale images and computed the histogram of the grayscale images for screening the poorly exposed images. For an underexposed image, a large portion of the pixel values are close to 0 (which corresponds to black), whereas for an overexposed image, a large portion of the pixel values are close to 255 (which corresponds to white). We considered an image to

Figure 4.3: Label correlation in the EyePACS training set.

be underexposed if 90% of the pixel values are between 0 and 30. We consider an image to be overexposed if 80% of the pixel values of the image are between 225 and 255. The threshold is set differently since the retinal images contain some black borders. Note that there could be other ways to screen the poorly exposed image. For example, another equivalent method would be using the average pixel values of the grayscale image to screen the poorly exposed image. Our classification rule yields 145 underexposed images and 14 overexposed images in the training dataset. In the test set, our classification rule yields 190 underexposed images and 11 overexposed images. As the poorly exposed images comprised only 0.5% of the training set or test set, we did not remove any images from the training or test sets. An underexposed retinal image with its RGB histograms, a properly exposed image with its RGB histograms and an overexposed image with its RGB histograms are shown in

Figure 4.5.

Figure 4.4: Random sample images from the EyePACS dataset [5].

4.2 Image Augmentation and Preprocessing

As discussed in Section 4.1, the distribution of the DR label values in the EyePACS dataset is highly imbalanced and the sizes of the images vary from 400×315 pixels to 5184×3456 pixels. Therefore we first resize the size of all the images in our dataset to

Figure 4.5: Underexposed/Normal/Overexposed Images. Small figures with Red/Green/Blue color under each retinal image are the histograms of the Red/Green/Blue color channel.

a uniform value using bilinear interpolation. We resize all images to images with 672×672 pixels since some of the subtle differences between retinal images with different DR levels may not be captured by the images with a smaller size. To deal with the data imbalance problem, the state-of-the-art solutions for learning from imbalanced data include sampling methods (undersample the majority classes or oversample the minority classes to balance the dataset) and cost-sensitive learning [66] (assigning different weights to samples from different classes). As deep learning models usually have millions of parameters, a large amount of data is required for training such models. Undersampling will result in training models with limited data (i.e., some training samples from majority class are dropped) and cost-sensitive learning needs us to carefully pre-define the weights assigned to each class which may be time-consuming. Thus we choose to augment the minority classes using several image augmentation techniques to balance the training set.

We applied two types of image augmentation techniques to expand the train-

ing set. The first type includes geometric augmentation techniques which alter the geometry of the image by mapping the individual pixel values to new destinations [67]. To be more specific, we applied image flipping (horizontally or vertically), image rotation (0-360 degrees) and image cropping (90%). The second type of image augmentation technique is the Fancy PCA color augmentation proposed in [7] which alters the pixel values of RGB channels. The basic idea of Fancy PCA color augmentation is to alter the intensities of RGB channels of the image by adding small perturbations on those RGB channels. The perturbation for each image is first calculated by the eigenvectors of the covariance matrix of RGB channels multiplied by the corresponding eigenvalues scaled by a Gaussian random variable with a mean of zero and a standard deviation of 0.1. Then the RGB color perturbations are added to the RGB channels of the original image to form the augmented image. The Fancy PCA algorithm is summarized in Algorithm 4.1. A comparison of an original retinal image and the augmented retinal image using Fancy PCA is shown in Figure 4.6. Although the difference between these two images is not easily visible to the human eye, it has been shown that applying the Fancy PCA algorithm generally improves the model performance [7]. In order to generate an image, we first randomly apply one of the geometric augmentation techniques on the input image. Following that the augmented images are standardized by subtracting the channel means and dividing the channel standard deviations. Then we apply the Fancy PCA color augmentation technique to bring more varieties to the training images. We utilized the open source scikit-image library [68] to implement the discussed data augmentation techniques.

Figure 4.6: Illustration of the original image and the augmented image using Fancy PCA algorithm.

4.3 CNN Training

As illustrated in Figure 4.1, the proposed DR classification system is based on the ensemble of two computationally efficient CNN models, namely MobileNet-Dense and MobileNetV2. In this section, we describe the training details of the MobileNet-Dense model and MobileNetV2 model including the choice of the loss function and hyper-parameter tuning.

4.3.1 Evaluation Metric and Loss Function

Evaluation Metric

As the DR disease develops progressively, we consider the prediction of DR severity levels as an ordinal classification task [69] rather than an ordinary classification task. Ordinal classification is different from ordinary classification in the sense that the labels are ordered (e.g. disease levels from 0 to 4), while the labels for an ordinary classification task are not ordered (e.g. cat vs dog). Many metrics can be used to

Algorithm 4.1 Fancy PCA color augmentation algorithm [7].

Input:

An $m \times m \times 3$ standardized image $\mathbf{H}$;

- 1: Create an $m^2 \times 3$ matrix $\mathbf{H}_s$ given the $m \times m \times 3$ input image $\mathbf{H}$, where $\mathbf{H}_s[:, 1]$ contains all the red pixel data, $\mathbf{H}_s[:, 2]$ contains all the green pixel data and $\mathbf{H}_s[:, 3]$ contains all the blue pixel data;
- 2: Compute the covariance matrix of the $\mathbf{H}_s$.
- 3: Calculate the eigenvector $\mathbf{p}$ and eigenvalue λ of the covariance matrix.
- 4: Obtain the augmented RGB image $\mathbf{K}$ using:

$$[k_{i,j}^R, k_{i,j}^G, k_{i,j}^B]^T = [h_{i,j}^R, h_{i,j}^G, h_{i,j}^B]^T + [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3] [\alpha_1 \lambda_1, \alpha_2 \lambda_2, \alpha_3 \lambda_3]^T$$

- $\mathbf{p}_i$ is the i^{th} eigenvector of the 3×3 covariance matrix.
- λ_i is the i^{th} eigenvalue of the 3×3 covariance matrix.
- α_i is the i^{th} value of the Gaussian random variable with a mean of zero and a standard deviation of 0.1.
- $k_{i,j}^R, k_{i,j}^G, k_{i,j}^B$ are the ij^{th} entry of the Red/Green/Blue channel of the augmented image $\mathbf{K}$.
- $h_{i,j}^R, h_{i,j}^G, h_{i,j}^B$ are the ij^{th} entry of the Red/Green/Blue channel of the input image $\mathbf{H}$.

Output:

An $m \times m \times 3$ augmented RGB image $\mathbf{K}$;

evaluate classification performance such as accuracy, specificity, precision, recall, F1-score, etc. In this thesis, we adopt quadratic weighted kappa (QWK) [31] score as the evaluation matrix since QWK is designed to measure the agreement of two raters on labels with ordinal scales and it has been used for reporting DR classification performance for existing models [37, 32, 70, 39].

The QWK, k_w , is defined as follows:

$$k_w = 1 - \frac{\sum_{i,j} w_{i,j} o_{i,j}}{\sum_{i,j} w_{i,j} e_{i,j}} \quad (4.1)$$

The $\mathbf{W}$, $\mathbf{O}$ and $\mathbf{E}$ are $M \times M$ matrices which are used for calculating QWK score,

where M is the number of classes. The matrix $\mathbf{O}$ is the histogram matrix of the observed ratings, where $o_{i,j}$ is the number of images that received a rating i by rater A (human) and a rating j by rater B (the trained classifier). The matrix $\mathbf{W}$ is the weight matrix, where $w_{i,j}$ is the weights assigned to different types of disagreement and it is given by:

$$w_{i,j} = \frac{(i - j)^2}{(M - 1)^2} \quad (4.2)$$

The matrix $\mathbf{E}$ is a histogram matrix of expected ratings, the $e_{i,j}$ is calculated as the outer product between two rater's histogram vector of ratings (assuming that the rating process for rater A and rating process for rater B are independent of each other), as shown blow:

$$e_{i,j} = \frac{\sum_i o_{i,j} \sum_j o_{i,j}}{N} \quad (4.3)$$

where N is the total number of samples in the evaluation dataset. The N in the denominator is used for scaling the $e_{i,j}$ so that the $e_{i,j}$ and $o_{i,j}$ have the same scale.

Loss Function

After the evaluation metric is chosen, a loss function is needed to train our models. We can not use non-differentiable QWK as the loss function due to the fact that the backpropagation algorithm requires the loss function to be differentiable. For multi-class classification tasks, the categorical cross-entropy loss L is commonly used and it is given by:

$$L = -\frac{1}{N} \sum_i \sum_j t_{i,j} \log(p_{i,j}). \quad (4.4)$$

where $p_{i,j}$ represents predicted probability value of sample i to be in class j given by the neural network model, $t_{i,j}$ represents the ground truth probability value of sample i to be in class j (i.e., $t_{i,j} = [0, \dots, 1, \dots, 0]$ contains a single 1 at the j^{th} position if sample i are labeled as class j .), N represents total number of samples.

The categorical cross-entropy is not suitable for the ordinal DR classification task as it failed to consider the order of the labels and it only takes the predicted probability of the true class into consideration, which results in assigning same loss for different types of misclassification errors. To explain this, suppose there are two retinal images $\mathbf{I}_1$ and $\mathbf{I}_2$ with proliferative DR (i.e., the ground truth probability $\mathbf{t}_1 = \mathbf{t}_2 = [0, 0, 0, 0, 1]$). Suppose the predicted class probabilities (predicted by the model) for $\mathbf{I}_1$ and $\mathbf{I}_2$ are given by:

$$\mathbf{p}_1 \triangleq [p_{1,0}, p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}] = [0.9, 0, 0, 0, 0.1]$$

$$\mathbf{p}_2 \triangleq [p_{2,0}, p_{2,1}, p_{2,2}, p_{2,3}, p_{2,4}] = [0, 0, 0, 0.9, 0.1]$$

with $\mathbf{p}_1$, the model predicts that $\mathbf{I}_1$ has a 90% probability of being an image without DR and 10% probability of being an image with proliferative DR. With $\mathbf{p}_2$, the model predicts that $\mathbf{I}_2$ has a 90% probability of being an image with severe DR and 10% probability of being an image with proliferative DR. Although neither $\mathbf{p}_1$ nor $\mathbf{p}_2$ gives the correct prediction, $\mathbf{p}_2$ is better than $\mathbf{p}_1$ in the sense that the predicted severity level is closer to the ground truth severity level. We would expect the loss function to assign different loss values for $\mathbf{p}_1$ and $\mathbf{p}_2$, however the corresponding categorical cross-entropy loss values for $\mathbf{p}_1$ and $\mathbf{p}_2$ are the same, as shown below:

$$L_1 = -0 \times \log(0.9) - 0 \times \log(0) - 0 \times \log(0) - 0 \times \log(0) - 1 \times \log(0.1) = \log(10)$$

$$L_2 = -0 \times \log(0) - 0 \times \log(0) - 0 \times \log(0) - 0 \times \log(0.9) - 1 \times \log(0.1) = \log(10)$$

Following the 2nd place method [70] in Kaggle DR challenge, we consider the DR

classification problem as an ordinal regression problem and we use the MSE loss to train the CNN models as both MSE loss and QWK score introduce quadratic penalty to the disagreement between the predicted result and the ground truth label. We treat the categorical label (0,1,2,3,4) as continuous values and we minimize the MSE between the output of our CNN model and ground truth categorical label. Note that the output of our neural network is a real number and it is bounded between 0 and 4 (e.g., $y_{pred} = 3.67$). The categorical predictions are then obtained by thresholding the predicted real values. The threshold values are given by $[\eta_0 = 0, \eta_1, \eta_2, \eta_3, \eta_4, \eta_5 = 4]$. For any $\eta_i < y_{pred} < \eta_{i+1}$, we interpret y_{pred} to be class i . The optimal threshold values are searched using the gradient-free Powell's method [71] on the validation set and the optimal threshold values are applied on the test set to obtain the test QWK score.

4.3.2 Hyperparameter Tuning and Training

Hyperparameter Tuning

The MobileNet-Dense model and MobileNetV2 model for DR classification are summarized in Table 4.2 and Table 4.3 respectively. Compared to the standard MobileNet-Dense model and standard MobileNetV2 model discussed in Chapter 3. Two fully connected layers are added between the GlobalAvgPool Layer and the Output Layer to improve the performance at the cost of some added complexity.

Same as the experiment on CIFAR dataset, grid search are used to find the hyperparameters (t and K) for MobileNetV2-Dense model and hyperparameters (t and α) for MobileNetV2 model. The hyperparameter searching space is listed below:

$$t \in \{2, 3, 4, 5, 6\}$$

$$K \in \{16, 32, 48, 64\}$$

Table 4.2: The structure of MobileNet-Dense for DR classification

Input	Operator	M	r	s	K	t
$672^2 \times 3$	Conv3x3 Layer	32	1	2	-	-
$336^2 \times 32$	Dense Block	48	1	1	16	1
$336^2 \times 48$	Dense Module	96	2	2	48	3
$168^2 \times 96$	Conv1x1 Layer	48	1	1	-	-
$168^2 \times 48$	Dense Module	144	3	2	48	3
$84^2 \times 144$	Conv1x1 Layer	72	1	1	-	-
$84^2 \times 72$	Dense Module	216	4	2	48	3
$42^2 \times 216$	Conv1x1 Layer	108	1	1	-	-
$42^2 \times 108$	Dense Module	300	5	2	48	3
$21^2 \times 300$	Conv1x1 Layer	1280	1	1	-	-
$21^2 \times 1280$	GlobalAvgPool Layer	1280	1	-	-	-
1280	Fully Connected Layer	256	2	-	-	-
256	Output Layer	1	-	-	-	-

Table 4.3: The structure of MobileNetV2 for DR classification

Input	Operator	M	r	s	t
$672^2 \times 3$	Conv3x3 Layer	40	1	2	-
$336^2 \times 40$	Residual Module	24	1	1	1
$336^2 \times 24$	Residual Module	32	2	2	6
$168^2 \times 32$	Residual Module	40	3	2	6
$84^2 \times 40$	Residual Module	80	4	2	6
$42^2 \times 80$	Residual Module	128	3	1	6
$42^2 \times 128$	Residual Module	208	3	2	6
$21^2 \times 208$	Residual Module	416	1	1	6
$21^2 \times 416$	Conv1x1 Layer	1664	1	1	-
$21^2 \times 1664$	GlobalAvgPool Layer	1664	1	-	-
1664	Fully Connected Layer	256	2	-	-
256	Output Layer	1	-	-	-

$$\alpha \in \{1.0, 1.1, 1.2, 1.3\}$$

In order to speed up the hyperparameter tuning process, models with different hyperparameter settings are trained using images with a small size (i.e., 128×128 pixels) as it is pointed out in [72] that the optimal hyperparameter values are largely independent of the image size. We randomly sample 60% of the EyePACS training data to train and the hyperparameters set which yields a model with highest QWK on the validation set is chosen. The hyperparameter set we found for MobileNet-Dense and MobileNetV2 are $[t = 3, K = 48]$ and $[t = 6, \alpha = 1.3]$ respectively.

Training CNN models

The MobileNet-Dense and MobileNetV2 are trained using a batch size of 4 for 200 epochs. The initial learning rate is set to 10^{-4} , and it is reduced to 10^{-5} at 80% of the total number of training epochs. We trained each model for 200 epochs as we observed that training more than 250 epochs resulted in overfitting and 200 epochs are sufficient for the validation loss to converge. After each epoch, we evaluate the QWK score of our model on the validation set and the model checkpoints (i.e., The model checkpoint at epoch t stores the current values of all the learnable parameters) which achieve a QWK score higher than 0.810 are saved for further testing and model ensembling.

4.4 Ensemble Learning

Ensemble of machine learning models (also known as ensemble learning) generally improves the classification system's robustness and accuracy [73]. It mimics the common practice of making decisions based on the decisions of multiple experts. Ensemble learning has been widely applied in medical image classification tasks, such as cancer detection [74] and Alzheimer's Disease detection [75]. Therefore we applied

ensemble learning so as to improve the performance of the classification system. The ensemble learning process consists of three steps: the feature extraction step, the feature reduction step and the model stacking step.

4.4.1 Feature Extraction and Feature Blending

We utilized the label correlation for a pair of eyes by training classifiers to screen DR using features from both of the eyes as it has been shown in [70, 39] that utilizing the label correlation generally improves the overall performance. To be more specific, the features of each retinal image are extracted from the last fully-connected layer of the trained MobileNet-Dense model and the trained MobileNetV2 model (the feature of each retinal image is a 256×1 matrix since the last fully connected layer contains 256 nodes). Then the features of the left eye and the right eye are concatenated. The concatenated feature is given by:

$$\mathbf{F}_{\text{concatenated}} = \left[\mathbf{F}_{\text{MobileNet-Dense}}^{\text{left}}, \mathbf{F}_{\text{MobileNet-Dense}}^{\text{right}}, \mathbf{F}_{\text{MobileNetV2}}^{\text{left}}, \mathbf{F}_{\text{MobileNetV2}}^{\text{right}} \right]$$

where $\mathbf{F}_{\text{MobileNet-Dense}}^{\text{left}}$ and $\mathbf{F}_{\text{MobileNetV2}}^{\text{left}}$ denote the left eye's features extracted from MobileNet-Dense model and MobileNetV2 model respectively. Similarly, $\mathbf{F}_{\text{MobileNet-Dense}}^{\text{right}}$ and $\mathbf{F}_{\text{MobileNetV2}}^{\text{right}}$ denote the right eye's features extracted from MobileNet-Dense model and MobileNetV2 model respectively.

4.4.2 Feature Reduction

The PCA (Principal Component Analysis) feature reduction is performed on the concatenated features in order to reduce the dimensionality of the concatenated feature and thereby reducing the overfitting as well as speeding up the training process. We select the number of components such that 99% of the variance is explained.

4.4.3 Model Stacking

We used the stacking technique [76] to perform ensemble learning. Stacking is an ensemble learning technique that combines multiple base-level models via a meta-classifier. The base-level models (i.e. MobileNet-Dense and MobileNetV2) are first trained using the EyePACS DR training set, then the meta-classifier is trained using the features extracted from the base-level models. To be more specific, we trained one FCNN as the meta-classifiers to screen left eyes using concatenated features and we trained another FCNN as the meta-classifiers to screen right eyes. We used the same structure for these two FCNNs. The first hidden layer of the FCNN contains 256 hidden nodes and the second hidden layer contains 128 hidden nodes. The output layer contains one node and the MSE loss is used for training. A weight decay of 0.025 is applied to reduce overfitting. The FCNNs are trained using a batch size of 256 for 120 epochs. The initial learning rate is set to 0.0005, and it is successively decreased by 10 at 40%, 60% and 80% of the total number of training epochs.

4.5 Summary

In this chapter, we proposed an automatic DR classification system based on the ensembling of two computationally efficient CNN models, namely the MobileNet-Dense model and the MobileNetV2 model. The system is designed following the typical workflow described in Section 2.3. Exploratory data analysis on EyePACS training set is first described followed by the image augmentation techniques used to address the data imbalance problem. Next, hyperparameter optimization and training details of the MobileNet-Dense and MobileNetV2 models are discussed. Lastly, the ensemble learning that was used is described.

Chapter 5

Performance Evaluation of the Diabetic Retinopathy Classification System

In this Chapter, we present the performance evaluation of the automatic DR classification system proposed in Chapter 4. We demonstrate the effectiveness of the proposed DR classification system using two independent DR dataset, namely the EyePACS dataset and the Messidor database. In Section 5.1, the performance of the proposed DR classification system is compared with some state-of-the-art methods [39, 37] on the EyePACS dataset. In Section 5.2, the performance of the proposed DR classification system is compared with some state-of-the-art methods [36, 37] on the Messidor database [9]. A brief summary is provided in Section 5.3.

5.1 Performance Evaluation on the EyePACS Test Set

In this section, we present the performance evaluation of the proposed system using the EyePACS test set. The overall performance is evaluated using the QWK score [31], the MAdds and the Parameters. The classification performance for each DR severity class is measured using the precision, recall and F1-Score [77].

Table 5.1: Performance comparison on EyePACS test set

Method	Test QWK	Parameters ¹	MAdds ²	Testing Time ³
Method in [39] ⁴	0.849	11.8M	45.0B	-
Zoom-In-Net [37]	0.854	-	-	-
MobileNet-Dense	0.825	1.8M	3.7B	25.6ms
MobileNetV2	0.822	4.2M	4.6B	28.7ms
Model Ensemble (1+1) ⁵	0.851	6.2M	8.3B	54.3ms
Model Ensemble (2+1) ⁵	0.852	8.0M	12.0B	79.9ms
Model Ensemble (1+2) ⁵	0.852	10.4M	12.9B	83.0ms
Model Ensemble (2+2) ⁵	0.852	12.3M	16.6B	108.6ms

¹ Parameters is in Millions.

² MAdds is in Billions and it is the MAdds for predicting one image.

³ Test time is in milliseconds and it is the inference time for predicting one image.

⁴ Method in [39] is the top-ranked method in the Kaggle DR Detection Challenge, it ensembled three CNNs to obtain the final results. We use this method as the benchmark method.

⁵ Model Ensemble ($M+N$) refers to the ensemble of Top- M saved checkpoints of MobileNet-Dense and Top- N saved checkpoints of MobileNetV2.

5.1.1 The QWK score and Model Complexity

We use the QWK score, MAdds and Parameters to report the overall performance. The QWK score, MAdds and Parameters of the proposed system and those of some state-of-the-art models are presented in Table 5.1. It can be seen that MobileNet-Dense model achieves a QWK score of 0.825 on EyePACS test set which is slightly higher than the QWK score of 0.822 achieved by MobileNetV2 model, but using 57% fewer Parameters, 20% fewer MAdds and 11% lesser running time. Let Model Ensemble ($M+N$) to denote the ensemble of Top- M saved checkpoints of MobileNet-Dense and Top- N saved checkpoints of MobileNetV2. It can be seen that the same QWK score (0.852) is achieved by Model Ensemble (2+1), Model Ensemble (1+2) and Model Ensemble (2+2). Considering the MAdds and the Parameters, we select the result of Model Ensemble (2+1) as our final result to compare with existing works [39, 37].

As shown in Table 5.1, the Zoom-In-Net [37] method achieves the highest QWK score (0.854) on the EyePACS test set. Unfortunately, we are not able to estimate the Parameters or MAdds of the Zoom-In-Net based on the information provided in [37]. Therefore we selected the top-ranked method in the Kaggle DR challenge (i.e., Method [39] in Table 5.1) as the benchmark method. Our Model Ensemble (2+1) achieves a QWK score of 0.852 compared to a QWK score of 0.849 achieved by benchmark method [39] while using 32% fewer parameters and 73% fewer MAdds. The results indicate that our system is effective and efficient compared to the benchmark method.

5.1.2 Performance Evaluation for Each Class

As the QWK score measures the overall agreement between two raters' classification results, it is hard to interpret the model performance for each class using only the QWK score. Therefore, we also investigated the model performance of the proposed DR classification system for each DR severity class using the commonly used metrics of precision, recall and F1-score [77].

We first recall the concept of confusion matrix since the precision, recall and F1-score can be easily calculated using the confusion matrix. A confusion matrix $\mathbf{C}$ is a matrix in which rows represent the ground truth label and columns represents the classification result, as shown in Figure 5.1. The $c_{i,j}$ is the number of samples predicted as class j by the classifier with the ground truth label class i .

The precision, recall and F1-Score for class k are calculated as follows:

$$Precision = \frac{TP_k}{TP_k + FP_k} \quad (5.1)$$

$$Recall = \frac{TP_k}{TP_k + FN_k} \quad (5.2)$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (5.3)$$

where TP_k , FP_k , FN_k denote the number of true positives, false positives, and false negatives for class k respectively. TP_k is the number of samples correctly predicted as class k by the classifier, i.e., $c_{k,k}$. The FP_k is the number of samples wrongly predicted as class k by the classifier, i.e., $\sum_{i|i \neq k} c_{i,k}$. The FN_k is the number of samples wrongly predicted as not class k , i.e., $\sum_{j|j \neq k} c_{k,j}$.

As the confusion matrices of the state-of-the-art methods [39, 37] on EyePACS test set are not given, we only reported the precision, recall and F1-Score of our DR classification system. The confusion matrices of the predicted results on EyePACS test set are given in Figure 5.1 and the corresponding precision, recall and F1-Score are shown in Table 5.2.

It can be seen from Table 5.2 that the ensemble learning generally improves the classification performance of each class. Another noteworthy point is that our system achieves relatively low F1-score on screening Mild DR and Severe DR, one possible reason could be the categorical predictions are obtained by thresholding the output values in order to maximize the overall QWK score instead of maximizing the F1-score of each class. Thus adjusting the threshold may improve the F1-scores of these two classes.

(a) Confusion Matrix of MobileNetV2

(b) Confusion Matrix of MobileNet-Dense

(c) Confusion Matrix of Model Ensemble (2+1)

Figure 5.1: Confusion matrices of: (a) MobileNetV2, (b) MobileNet-Dense, (c) Model Ensemble (2+1)

Table 5.2: Per-class performance on EyePACS test set

Class	Precision			Recall			F1-score		
	MobV2 ¹	Mob-D ²	Ensemble ³	MobV2	Mob-D	Ensemble	MobV2	Mob-D	Ensemble
Healthy	0.92	0.92	0.94	0.93	0.94	0.94	0.93	0.93	0.94
Mild	0.28	0.28	0.31	0.39	0.40	0.40	0.32	0.33	0.35
Moderate	0.75	0.76	0.75	0.50	0.40	0.51	0.60	0.52	0.61
Severe	0.30	0.25	0.30	0.50	0.61	0.70	0.37	0.36	0.42
Proliferate	0.66	0.61	0.76	0.49	0.54	0.52	0.57	0.57	0.62

¹ The MobV2 denotes the MobileNetV2 model.

² The Mob-D denotes the MobileNet-Dense model.

³ The Ensemble denotes the Ensemble (2+1) .

5.2 Performance Evaluation on the Messidor Database

In this section, we test the performance of the proposed automatic DR classification system on another widely used independent diabetic retinopathy dataset, namely the Messidor database [9], to show the generalization ability of the proposed system. As the label distribution and image quality of the Messidor database are different from those of the EyePACS dataset, thus the basic EDA of Messidor database is first given followed by the comparison of our results with the state-of-the-art results [36, 78, 37].

5.2.1 EDA of the Messidor Database

The Messidor database consists of 1200 retinal images. The retinal images are in one of three sizes: 1440×960 , 2240×1488 , or 2304×1536 . A retinopathy grade is provided by an ophthalmologist for each retinal image in the Messidor database and the grade ranges from 0 to 3. Random sample retinal images with different DR grades from the Messidor database are shown in Figure 5.2. Each row contains five retinal images with the same DR grade. The distribution of DR grades in Messidor database is shown in Figure 5.3, with 46% of the images labeled as grade 0, 13% as grade 1, 20% as grade 2, and 21% as grade 3.

Figure 5.2: Random sample images from the Messidor database [9].

5.2.2 Performance Evaluation

As there are only 1200 images in Messidor database which is not adequate to train a CNN model, Holly et al.[36] suggested building classifiers using features extracted from the CNN models trained on other DR dataset such as EyePACS DR dataset. As the DR grade for the Messidor database ranges from 0 to 3 while the DR label value

Figure 5.3: Distribution of DR grades in the Messidor Database.

ranges from 0 to 4 for the EyePACS dataset, we adopt the same scheme in [36, 37] and perform two binary classification tasks (i.e., Referable versus Non-Referable, Normal versus Abnormal) to demonstrate the generalization ability. For Normal/Abnormal classification task, images with grade 0 are treated as normal and images with other grades are treated as abnormal. For Referable/Non-Referable classification task, images with grade 0 and grade 1 are treated as non-referable and images with grade 2 and grade 3 are treated as referable. The training schemes and the results for these two binary classification tasks are described below:

- For Normal/Abnormal classification task, following the scheme used in [36, 37], we trained a logistic classifier using the 256-dimensional features (i.e., the output of the last fully connected layer) extracted from the EyePACS training

set and tested the performance using the entire Messidor database. The AUC (Area Under The ROC curve) score and Acc (Accuracy) score are used as the evaluation metrics, as shown in Table 5.3. It can be seen that MobileNet-Dense model and MobileNetV2 model achieve same AUC score and same Acc score. Small improvements are obtained by the ensemble of MobileNet-Dense model and MobileNetV2 model. Specifically, the AUC score reached to 0.962 and the Acc score reached to 0.917. Results have shown that our system outperforms the Zoom-In-Net method in both AUC and Acc scores. The ROC curves of our methods are shown in Figure 5.4.

- For Referable/Non-Referable classification task, following the scheme used in [36, 37], 10-fold cross-validation is performed on the entire Messidor database (i.e., The dataset is randomly partitioned into 10 subsets, with same number of samples in each subset. A single subset is retained as the test data for testing the model, and the remaining 9 subsets are used as training data. The cross-validation process is repeated 10 times, with each of the 10 subsets used once as the retained test data). A logistic regression model is trained to perform the binary classification task. The mean AUC score and mean Acc score are used as the evaluation metrics, as shown in Table 5.4. It can be seen that the mean AUC scores achieved by MobileNet-Dense and MobileNetV2 are 0.967 and 0.970 respectively. The mean Acc scores achieved by MobileNet-Dense and MobileNetV2 are 0.922 and 0.923 respectively. A small improvement on mean Acc score (from 0.923 to 0.924) is obtained by the ensemble of MobileNet-Dense model and the MobileNetV2 model. Results have shown that our system outperforms the Zoom-In-Net method in both mean AUC and mean Acc scores. The ROC curves of our methods are shown in Figure 5.5.

Figure 5.4: ROC curves for Normal/Abnormal task.

Figure 5.5: ROC curves for Referable/Non-Referable task.

Table 5.3: Performance comparison on Normal/Abnormal screening task

Method	AUC	Acc
VNXK [36]	0.870	0.871
CKML [36]	0.862	0.858
Human Expert A [78]	0.922	-
Human Expert B [78]	0.865	-
Zoom-In-Net [37]	0.921	0.905
MobileNet-Dense	0.959	0.908
MobileNetV2	0.959	0.908
Model Ensemble (1+1)	0.962	0.917

Table 5.4: Performance comparison on Referable/Non-Referable screening task

Method	mean AUC	mean Acc
VNXK [36]	0.887	0.893
CKML [36]	0.891	0.897
Human Expert A [78]	0.94	-
Human Expert B [78]	0.92	-
Zoom-In-Net [37]	0.957	0.911
MobileNet-Dense	0.967	0.922
MobileNetV2	0.970	0.923
Model Ensemble (1+1)	0.970	0.924

In summary, our method outperforms the state-of-the-art Zoom-In-Net method on both Referable/Non-Referable screening and Normal/Abnormal screening tasks. These results indicate that our proposed methods have good generalization abilities for performing DR screening using retinal images came from different sources.

5.3 Summary

In this Chapter, the performance of our automatic DR classification system are presented. Results are obtained using the EyePACS test set and the Messidor database. For the EyePACS test set, it is found that the proposed system achieves a QWK score

of 0.852 compared to a QWK score of 0.849 achieved by the benchmark method [39], but using 32% fewer parameters and 73% fewer MAdds. For the Messidor database, our system outperforms the state-of-the-art method [37] on both Normal/Abnormal and Referable/Non-Referable screening tasks. The results indicate that our system is effective, efficient and have good generalization abilities for screening retinal images.

Chapter 6

Conclusion

In this chapter, we summarize the main contributions. Some possible future works are outlined as well.

6.1 Main Contributions

- In Chapter 3, we proposed a novel MobileNet-Dense model which is based on the DenseNet model [8] and the MobileNetV2 model [4]. We demonstrated its effectiveness and efficiency using the CIFAR-10 and CIFAR-100 datasets. The MobileNet-Dense and the MobileNetV2 achieve similar Acc on both datasets, but MobileNet-Dense uses 62% fewer parameters and 36% fewer MAdds.
- In Chapter 4, we proposed an automatic DR classification system based on the ensemble of two computationally efficient CNN models, namely the proposed MobileNet-Dense model and the existing MobileNetV2 model [4]. The system is trained using the EyePACS training set.
yu
- In Chapter 5, we evaluated the proposed DR classification system on the EyePACS dataset and the Messidor database. On the EyePACS dataset, the top-ranked method [39] in the Kaggle DR challenge was chosen as the benchmark method. Our system achieves a test QWK score of 0.852 compared to a test QWK score of 0.849 achieved by the benchmark method while using 32%

fewer parameters and 73% fewer MAdds. For the experiment on the Messidor database, our system achieves an AUC of 0.962 and an Acc of 0.917 compared to an AUC of 0.921 and an Acc of 0.905 achieved by the state-of-the-art method [37] on Normal/Abnormal screening task. On the Referable/Non-Referable screening task, our system achieves a mean AUC of 0.970 and a mean Acc of 0.924 compared to a mean AUC of 0.957 and a mean Acc of 0.911 achieved by the state-of-the-art method [37].

6.2 Future Work

Some possible extensions of our research work are listed below:

- **Optimizing the CNN architecture:** We proposed a novel MobileNet-Dense model which is constructed using 4 dense modules as the default setting. A shallower MobileNet-Dense model (e.g., a MobileNet-Dense model with 3 dense modules) may achieve similar performance with fewer MAdds and Parameters or a deeper MobileNet-Dense model may achieve noticeably better performance at an acceptable cost.
- **Pruning the trained model:** Another possible extension of our work would be further improving the computational efficiency of our DR classification system by pruning and retraining the trained MobileNet-Dense model and MobileNetV2 model (removing the convolution filters in convolutional layer or weights in fully connected layers which are less correlated to the final results) as it has been shown that network pruning and retraining reduce the number of MAdds and Parameters of the model while maintaining similar performance in various applications or benchmark datasets [79, 80].

- **Enhancing the image contrast:** As mentioned in Section 4.1, the EyePACS dataset contains some poorly exposed images. It would be interesting to investigate whether the overall classification performances of our system could be significantly improved by enhancing the image contrast using adaptive histogram equalization algorithm or contrast limited adaptive histogram equalization algorithm [81].
- **Stacking MobileNet-Dense with different CNN models:** As both MobileNetV2 and MobileNet-Dense use the depthwise separable convolutional layer as the basic building block and these two models achieve similar performance on both EyePACS dataset and Messidor database. Thus the features extracted from these two models might be highly correlated which results in limited improvement when performing ensemble learning. It would be interesting to investigate whether we can improve the performance of the DR classification system by stacking the MobileNet-Dense model with other types of computationally efficient CNN models which are not constructed using depthwise separable convolutional layers (e.g., SqueezeNet [51]).

Bibliography

- [1] “Indian Diabetic Retinopathy Image Dataset (IDRiD) website,” <https://idrid.grand-challenge.org/grading/>, accessed: 2018-05-30.
- [2] K. He *et al.*, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.
- [3] A. G. Howard *et al.*, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [4] M. Sandler *et al.*, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 4510–4520.
- [5] “Kaggle Diabetic Retinopathy Detection,” <https://www.kaggle.com/c/diabetic-retinopathy-detection/data>, accessed: 2018-04-30.
- [6] T. Kauppi *et al.*, “The diaretddb1 diabetic retinopathy database and evaluation protocol.” in *Proceedings of Medical Image Understanding and Analysis (MIUA)*, vol. 1, 2007, pp. 1–10.
- [7] A. Krizhevsky *et al.*, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 2012, pp. 1097–1105.
- [8] G. Huang *et al.*, “Densely connected convolutional networks,” in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2261–2269.
- [9] E. Decenciere *et al.*, “Feedback on a publicly distributed image database: the Messidor database,” *Image Analysis and Stereology*, vol. 33, no. 3, pp. 231–234, 2014.
- [10] F. Chentli, S. Azzoug, and S. Mahgoun, “Diabetes mellitus in elderly,” *Indian journal of endocrinology and metabolism*, vol. 19, no. 6, p. 744, 2015.
- [11] D. M. Nathan, “Long-term complications of diabetes mellitus,” *New England Journal of Medicine*, vol. 328, no. 23, pp. 1676–1685, 1993.

- [12] Y. Zheng *et al.*, “The worldwide epidemic of diabetic retinopathy,” *Indian Journal of Ophthalmology*, vol. 60, no. 5, pp. 428–431, 2012.
- [13] “Facts about diabetic eye disease,” <https://nei.nih.gov/health/diabetic/retinopathy>, accessed: 2019-03-20.
- [14] Early Treatment Diabetic Retinopathy Study Research Group *et al.*, “Classification of diabetic retinopathy from fluorescein angiograms: ETDRS report number 11,” *Ophthalmology*, vol. 98, no. 5, pp. 807–822, 1991.
- [15] E. Chew *et al.*, “American academy of ophthalmology retina panel: Preferred practice patterns,” *American Academy of Ophthalmology*, 2003.
- [16] C. I. Sánchez *et al.*, “Retinal image analysis to detect and quantify lesions associated with diabetic retinopathy,” in *Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, vol. 1. IEEE, 2004, pp. 1624–1627.
- [17] V. Esmann *et al.*, “Types of exudates in diabetic retinopathy,” *Acta Medica Scandinavica*, vol. 174, no. 3, pp. 375–384, 1963.
- [18] R. Klein *et al.*, “The Wisconsin epidemiologic study of diabetic retinopathy: VII. diabetic nonproliferative retinal lesions,” *Ophthalmology*, vol. 94, no. 11, pp. 1389–1400, 1987.
- [19] J. Kaur and D. H. Sinha, “Automated detection of diabetic retinopathy using fundus image analysis,” *International Journal of computer Science and Information Technologies*, vol. 3, p. 4794, 2012.
- [20] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [21] H. Bay *et al.*, “Surf: Speeded up robust features,” in *Proceedings of European Conference on Computer Vision (ECCV)*. Springer, 2006, pp. 404–417.
- [22] C. Geng and X. Jiang, “Face recognition based on the multi-scale local image structures,” *Pattern Recognition*, vol. 44, no. 10-11, pp. 2565–2575, 2011.
- [23] L. J. Zhi *et al.*, “Medical image retrieval using SIFT feature,” in *Proceedings of International Congress on Image and Signal Processing (CISP)*. IEEE, 2009, pp. 1–4.
- [24] C. Sinthanayothin *et al.*, “Automated detection of diabetic retinopathy on digital fundus images,” *Diabetic Medicine*, vol. 19, no. 2, pp. 105–112, 2002.

- [25] A. Singalavanija *et al.*, “Feasibility study on computer-aided screening for diabetic retinopathy,” *Japanese Journal of Ophthalmology*, vol. 50, no. 4, pp. 361–366, 2006.
- [26] S. C. Lee *et al.*, “Computer classification of nonproliferative diabetic retinopathy,” *Archives of Ophthalmology*, vol. 123, no. 6, pp. 759–764, 2005.
- [27] P. Kahai *et al.*, “A decision support framework for automated screening of diabetic retinopathy,” *International Journal of Biomedical Imaging*, vol. 2006, pp. 1–8, 2006.
- [28] B. Antal *et al.*, “An ensemble-based system for microaneurysm detection and diabetic retinopathy grading,” *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1720–1726, 2012.
- [29] H. Pratt *et al.*, “Convolutional neural networks for diabetic retinopathy,” *Procedia Computer Science*, vol. 90, pp. 200–205, 2016.
- [30] S. izza Rufaida and M. I. Fanany, “Residual convolutional neural network for diabetic retinopathy,” in *Proceedings of IEEE International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 2017, pp. 367–374.
- [31] J. Cohen, “Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit,” *Psychological Bulletin*, vol. 70, no. 4, pp. 213–220, 1968.
- [32] D. Zhang *et al.*, “Diabetic retinopathy classification using deeply supervised ResNet,” in *Proceedings of IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computed, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2017.
- [33] M. Alban and T. Gilligan, “Automated detection of diabetic retinopathy using fluorescein angiography photographs,” *Report of Standford Education*, 2016.
- [34] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 1–9.
- [35] S. Suriyal *et al.*, “Mobile assisted diabetic retinopathy detection using deep neural network,” in *Proceedings of Global Medical Engineering Physics Exchanges/Pan American Health Care Exchanges (GMEPE/PAHCE)*. IEEE, 2018, pp. 1–4.

- [36] H. H. Vo and A. Verma, “New deep neural nets for fine-grained diabetic retinopathy recognition on hybrid color space,” in *Proceeding of the IEEE International Symposium on Multimedia (ISM)*. IEEE, 2016, pp. 209–215.
- [37] Z. Wang *et al.*, “Zoom-in-net: Deep mining lesions for diabetic retinopathy detection,” in *Proceeding of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2017, pp. 267–275.
- [38] X. Zhang and O. Chutatape, “A SVM approach for detection of hemorrhages in background diabetic retinopathy,” in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN)*, vol. 4. IEEE, 2005, pp. 2435–2440.
- [39] B. Graham, “Competition report of Minpooling,” <https://www.kaggle.com/c/diabetic-retinopathy-detection/discussion/15801>, 2015, online; accessed 30 January 2019.
- [40] S. Agatonovic-Kustrin and R. Beresford, “Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research,” *Journal of Pharmaceutical and Biomedical Analysis*, vol. 22, no. 5, pp. 717–727, 2000.
- [41] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.
- [42] S. Shi and X. Chu, “Speeding up convolutional neural networks by exploiting the sparsity of rectifier units,” *arXiv preprint arXiv:1704.07724*, 2017.
- [43] D. E. Rumelhart *et al.*, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
- [44] C. Davis, “The norm of the schur product operation,” *Numerische Mathematik*, vol. 4, no. 1, pp. 343–344, 1962.
- [45] Y. N. Dauphin *et al.*, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization,” in *Proceeding of Neural Information Processing Systems (NIPS)*, 2014, pp. 2933–2941.
- [46] O. M. Parkhi *et al.*, “Deep face recognition,” in *Proceedings of the British Machine Vision Conference (BMVC)*, vol. 41, no. 3. British Machine Vision Association, 2015, pp. 1–12.
- [47] R. Girshick *et al.*, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 580–587.

- [48] I. Goodfellow *et al.*, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [49] M. Abadi, A. Agarwal, P. Barham *et al.*, “TensorFlow: Large-Scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: <http://tensorflow.org/>
- [50] P. Molchanov *et al.*, “Pruning convolutional neural networks for resource efficient inference,” *arXiv preprint arXiv:1611.06440*, 2016.
- [51] F. N. Iandola *et al.*, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.
- [52] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [53] Z. Liu *et al.*, “Learning efficient convolutional networks through network slimming,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2736–2744.
- [54] L. Sifre and S. Mallat, “Rigid-motion scattering for image classification,” Ph.D. dissertation, Citeseer, 2014.
- [55] Y. LeCun *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [56] L. Perez and J. Wang, “The effectiveness of data augmentation in image classification using deep learning,” *arXiv preprint arXiv:1712.04621*, 2017.
- [57] L. Rokach, “Ensemble-based classifiers,” *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.
- [58] K. He and J. Sun, “Convolutional neural networks at constrained time cost,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 5353–5360.
- [59] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *arXiv preprint arXiv:1505.00387*, 2015.
- [60] G. Pleiss *et al.*, “Memory-efficient implementation of Densenets,” *arXiv preprint arXiv:1707.06990*, 2017.
- [61] J. T. Springenberg *et al.*, “Striving for simplicity: The all convolutional net,” *arXiv preprint arXiv:1412.6806*, 2014.
- [62] C.-Y. Lee *et al.*, “Deeply-supervised nets,” in *Proceedings of Machine Learning Research (MLR)*, 2015, pp. 562–570.

- [63] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [64] C. H. Yu, “Exploratory data analysis,” *Methods*, vol. 2, pp. 131–160, 1977.
- [65] S. Dodge and L. Karam, “Understanding how image quality affects deep neural networks,” in *Proceedings of 8th International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2016, pp. 1–6.
- [66] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2008.
- [67] L. Taylor and G. Nitschke, “Improving deep learning using generic data augmentation,” *arXiv preprint arXiv:1708.06020*, 2017.
- [68] S. Van der Walt *et al.*, “scikit-image: Image processing in Python,” *PeerJ*, vol. 2, p. e453, 2014.
- [69] C. Winship and R. D. Mare, “Regression models with ordinal variables,” *American sociological review*, pp. 512–525, 1984.
- [70] A. Mathis and B. Stephan, “Competition report of TeamoO,” <https://www.kaggle.com/c/diabetic-retinopathy-detection/discussion/15617>, 2015, online; accessed 30 January 2019.
- [71] M. J. Powell, “An efficient method for finding the minimum of a function of several variables without calculating derivatives,” *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.
- [72] T. Hinz *et al.*, “Speeding up the hyperparameter optimization of deep convolutional neural networks,” *International Journal of Computational Intelligence and Applications*, p. 1850008, 2018.
- [73] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.
- [74] X. Yuan *et al.*, “A regularized ensemble framework of deep learning for cancer detection from multi-class, imbalanced training data,” *Pattern Recognition*, vol. 77, pp. 160–172, 2018.
- [75] A. Ortiz *et al.*, “Ensembles of deep learning architectures for the early diagnosis of the alzheimer’s disease,” *International Journal of Neural Systems*, vol. 26, no. 07, p. 1650025, 2016.
- [76] D. H. Wolpert, “Stacked generalization,” *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

- [77] D. M. Powers, “Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation,” 2011.
- [78] C. I. Sánchez *et al.*, “Evaluation of a computer-aided diagnosis system for diabetic retinopathy screening on public data,” *Investigative Ophthalmology and Visual Science*, vol. 52, no. 7, pp. 4866–4871, 2011.
- [79] S. Han *et al.*, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *arXiv preprint arXiv:1510.00149*, 2015.
- [80] H. Li *et al.*, “Pruning filters for efficient convnets,” *arXiv preprint arXiv:1608.08710*, 2016.
- [81] S. M. Pizer *et al.*, “Adaptive histogram equalization and its variations,” *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.