DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2020

Deep Learning Approach for Diabetic Retinopathy Grading with Transfer Learning

LINDA ANDERSEN

PHILIP ANDERSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Bachelor in Computer Science
Date: June 8, 2020
Supervisor: Alexander Kozlov
Examiner: Pawel Herman
Swedish title: Användning av Djupinlärning med Överföringsinlärning för gradering av Diabetisk Näthinnesjukdom
School of Electrical Engineering and Computer Science

Abstract

Diabetic retinopathy (DR) is a complication of diabetes and is a disease that affects the eyes. It is one of the leading causes of blindness in the the Western world. As the number of people with diabetes grows globally, so does the number of people affected by diabetic retinopathy. This demand requires that better and more effective resources are developed in order to discover the disease in an early stage which is key to preventing that the disease progresses into more serious stages which ultimately could lead to blindness, and streamline further treatment of the disease. However, traditional manual screenings are not enough to meet this demand. This is where the role of computer-aided diagnosis comes in. The purpose of this report is to investigate how a convolutional neural network together with transfer learning can perform when trained for multiclass grading of diabetic retinopathy. In order to do this, a pre-built and pre-trained convolutional neural network from Keras was used and further trained and fine-tuned in Tensorflow on a 5-class DR grading dataset. Twenty training sessions were performed and accuracy, recall and specificity were evaluated in each session. The results show that testing accuracies achieved were in the range of 35% to 48.5%. The average testing recall achieved for class 0, 1, 2, 3 and 4 was 59.7%, 0.0%, 51.0%, 38.7% and 0.8%, respectively. Furthermore, the average testing specificity achieved for class 0, 1, 2, 3 and 4 was 77.8%, 100.0%, 62.4%, 80.2% and 99.7%, respectively. The average recall of 0.0% and average specificity of 100.0% for class 1 (mild DR) were obtained because the CNN model never predicted this class.

Sammanfattning

Diabetisk näthinnesjukdom (DR) är en komplikation av diabetes och är en sjukdom som påverkar ögonen. Det är en av de största orsakerna till blindhet i västvärlden. Allt eftersom antalet människor med diabetes ökar, ökar även antalet med diabetisk näthinnesjukdom. Detta ställer högre krav på att bättre och effektivare resurser utvecklas för att kunna upptäcka sjukdomen i ett tidigt stadie, vilket är en förutsättning för att förhindra vidareutveckling av sjukdomen som i slutändan kan resultera i blindhet, och att vidare behandling av sjukdomen effektiviseras. Här spelar datorstödd diagnostik en viktig roll. Syftet med denna studie är att undersöka hur ett faltningsnätverk, tillsammans med överföringsinformation, kan prestera när det tränas för multiklass gradering av diabetisk näthinnesjukdom. För att göra detta användes ett färdigbyggt och färdigtränat faltningsnätverk, byggt i Keras, för att fortsättningsvis tränas och finjusteras i Tensorflow på ett 5-klassigt DR dataset. Totalt tjugo träningssessioner genomfördes och noggrannhet, sensitivitet och specificitet utvärderades i varje sådan session. Resultat visar att de uppnådda noggranheterna låg inom intervallet 35% till 48.5%. Den genomsnittliga testsensitiviteten för klass 0, 1, 2, 3 och 4 var 59.7%, 0.0%, 51.0%, 38.7% respektive 0.8%. Vidare uppnåddes en genomsnittlig testspecificitet för klass 1, 2, 3 och 4 på 77.8%, 100.0%, 62.4%, 80.2% respektive 99.7%. Den genomsnittliga sensitiviteten på 0.0% samt den genomsnittliga specificiteten på 100.0% för klass 1 (mild DR) erhölls eftersom CNN modellen aldrig förutsåg denna klass.

Abbreviations

ANN	Artificial Neural Network
CAD	Computer-Aided Diagnosis
CNN	Convolutional Neural Network
DL	Deep Learning
DR	Diabetic Retinopathy
IDRiD	Indian Diabetic Retinopathy image Dataset
ML	Machine Learning
NPDR	Non-Proliferative Diabetic Retinopathy
PDR	Proliferative Diabetic Retinopathy
WHO	World Health Organization

Contents

Chapter 1

Introduction

This chapter presents an introduction to the chosen field of study. Firstly, an explanation of diabetic retinopathy is given and what the current problem is in terms of diagnosis of the disease. Then, the research question is presented as well as the scope and delimitation of the study and an explanation of the approach for the study. Lastly, a section about how the thesis is organized follows.

1.1 Diabetic Retinopathy

Diabetes is according to the World Health Organization (WHO) an increasingly rising problem around the world. One common complication of long-term diabetes is diabetic retinopathy (DR) which is an eye disease that could lead to vision loss or in the worst case blindness [1]. DR is one of the major causes of blindness in the Western world [2], and it is expected to increase as global prevalence of diabetes increases [3].

DR is diagnosed by analyzing different types of lesions, an area of abnormal tissue change, on a fundus image. A fundus image shows the rear of an eye and contains the macula, optic disc and retina [4]. In the early stages of DR will no treatment occur but the patients eyes will be regularly checked. If the eye disease worsens, then laser treatment is considered [5].

Early detection of DR in fundus images is crucial in order to prevent further development of the disease. As a matter of fact, early detection of DR can prevent blindness in 90% of cases [6]. However, manual screening for early detection of DR relies on experienced readers and is both time and labor intensive [7]. Therefore, automated methods for diagnosis of DR can reduce the workload on readers and improve further follow-up management with diabetic patients [8].

Many deep learning (DL) algorithms have in the recent years been developed for various tasks to analyse retinal fundus images, to develop computer-aided diagnosis (CAD) systems for DR. A commonly used DL architecture for DR-detection is the convolutional neural network (CNN). To properly learn a CNN requires a very large amount of data, but these large amounts of data are not available in the medical field, particularly for DR. Therefore, one solution is to use transfer learning, meaning that the CNN model is first taught in an end-to-end manner using dataset from a related domain and then fine-tuned using data from the specific domain [4].

With this motivation, the purpose of this report is to research the efficiency of DR grading classification with transfer learning.

1.2 Research Question

The study aims to investigate the following:

What is the performance of a convolutional neural network for diabetic retinopathy grading classification in fundus images with the use of transfer learning?

By performance, we mean accuracy, recall (sensitivity) and specificity which are the main measurements used in medicine together with machine learning [9]. Transfer learning, meaning the usage of a pre-trained network which will be further trained and fine-tuned for our specific research area, will be performed because of limited data and to save time in the learning process of the CNN. A histogram equalization algorithm will also be applied to the fundus images in the the pre-processing step for image contrast enhancement [10].

1.3 Scope/Delimitation

The scope of this study will be to explore the performance of DR grading classification with one pre-trained CNN architecture, namely Inception-v3. The CNN will be trained on one dataset, the Indian Diabetic Retinopathy Image Dataset (IDRiD), which consists of 516 images.

The fact that only one dataset was used, one which is representative for the Indian population, is a delimitation in the sense that this study may not be generalizable to other populations.

1.4 Approach

The study will be performed using a pre-trained CNN architecture called Inception-v3, which will be trained on IDRiD. The CNN will be fine-tuned, trained and evaluated using the software library Tensorflow together with the programming language Python. The pre-trained CNN will be imported from Keras, which is a Python DL API. In the pre-processing step, the images from the dataset will be cropped, resized and then a histogram equalization algorithm for image contrast enhancement will be applied. Finally, the images will be normalized to make the images on the same scale. The images will then be passed to the CNN for training. The CNN will be trained twenty times to evaluate how it performs on average, giving more representable results.

1.5 Outline

In this paper, we introduce a deep learning based CNN method for the task of grading DR in retinal fundus imagery. We then analyze the performance of the network.

The remainder of this paper is organized as follows: Chapter 2 presents a background to our study. First diabetic retinopathy is described, as well as what fundus imaging is. Then, a section on machine learning and deep learning will be given including CNNs. Then, a section on transfer learning will be presented. After this, a section on performance metrics will be given. Finally, this chapter will end by giving an overview of related work. Chapter 3 describes the chosen CNN architecture and methods used in building, training, evaluation of the network. Chapter 4 presents the results from our experiment. Chapter 5 and chapter 6 concludes the study with discussion on the results and future work.

Chapter 2

Background

This chapter will first present an overview of what diabetic retinopathy is and its characteristics. After this, a section about fundus imaging follows. Then a brief overview of machine learning and deep neural networks, with focus on convolutional neural networks will be given. Following this is a section about transfer learning and different performance metrics that are relevant in machine learning. The chapter will end in a section about previous work relevant to this study.

2.1 Characteristics of Diabetic Retinopathy

Diabetic retinopathy (DR) can be divided into two stages: non-proliferative diabetic retinopathy (NPDR) and advanced, proliferative diabetic retinopathy (PDR). NPDR can be sub-classified into mild, moderate and severe NPDR. Mild NPDR is characterized by increased vascular permeability. Moderate and severe NPDR is characterized by vascular closure. PDR is characterized by the growth of new, abnormal blood cells on the retina and posterior surface of the vitreous. Macular edema, characterized by retinal thickening from leaky blood vessels, can develop at all stages of DR [11][12].

DR is one of the leading causes of vision loss globally. Diagnosis of DR in diabetic patients in an early stage is key to preventing the disease to progress into more serious stages which ultimately could end in blindness. Therefore, diabetic patients regularly have their retinas examined. This is called diabetic retinopathy screening, and in such a screening, a trained reader examines fundus images of the patient in order to find early signs of the disease and then decides whether the patient needs to get a referral to an ophthalmologist for treatment. Various image analysis algorithms have been developed over the past decades in order to reduce the workload on human interpretation [13].

2.2 Fundus Imaging

Fundus imaging is the process where the interior surface of the eye, or fundus, is being photographed [4]. It is defined as "the process whereby a 2-D representation of the 3-D retinal semi-transparent tissues projected onto the imaging plane is obtained using reflected light", according to [14].

One of the most obvious application areas in which retinal screening application is used is for detection of retinal diseases, such as detection of glaucoma, age-related macular degeneration and retinopathy of prematurity and DR.

Early detection of DR through population screening, combined with the right treatment at the right time, has proven to prevent visual loss and blindness in patients with retinal complications of diabetes [15][16].

Below, a retinal fundus image from the dataset used in this report is shown.

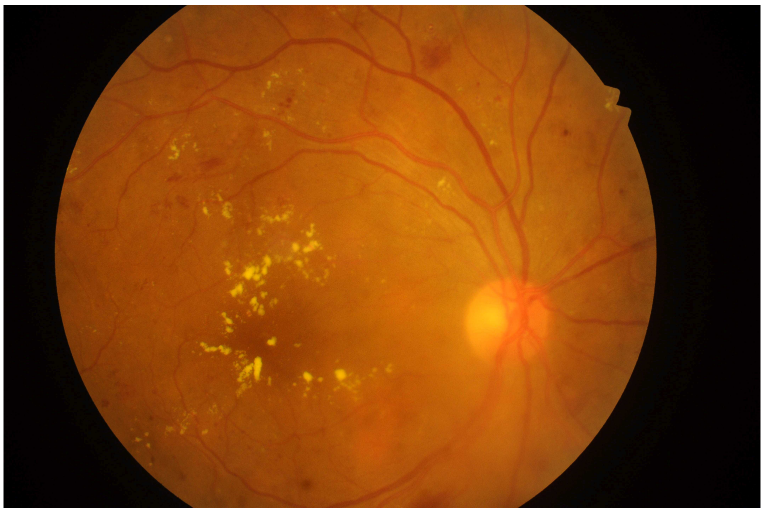

Figure 2.1: Retinal fundus image from the IDRiD dataset, see section 3.1.

2.3 Machine Learning

Machine learning (ML) is a subfield of artificial intelligence and can be described as the technique of programming computers in order to optimize a performance criterion using example data or past experience. A model is defined up to some parameters, and learning is the process where an algorithm is executed in which these parameters are optimized using training data or past experience [17].

ML is used in various application areas, such as retail, finance, manufacturing and medicine. In fact, ML algorithms were from the initial beginning designed and used for analyzation of medical datasets [18]. For example, banks use past data in order to build ML models to use in credit applications, fraud detection and stock market. In medicine, ML is used for medical diagnostics.

There exist different learning techniques in ML, meaning ways in which the ML model learns. One common learning technique used is supervised learning. In supervised learning, the training data include both input data and the desired output (the correct classification, or labels) data, which is not the case with other learning techniques [19]. Supervised learning is the learning technique used in this report.

2.3.1 Deep Learning and Neural Networks

An important subfield of ML which has gotten a lot of attention in the last decade is deep learning (DL). DL is a concept taken from animal brains based on so called neurons and neural networks. In the field of computer science, these are referred to as artificial neurons or nodes and artificial neural networks (ANNs). An ANN consists of an input layer, one or more hidden layers and an output layer where a layer is multiple nodes grouped together. Between the layers there are connections with weights and it is by updating the weights an ANN learns. One advantage of ANNs over other ML algorithms is that ANNs have automatic feature extraction, where a feature is either a measurable attribute or characteristic in a dataset. Selecting relevant features from a dataset is crucial for prediction accuracy and to decrease learning time because it removes redundant data and reduces the problem of overfitting, which occurs when a ML model is trained too closely on a dataset making generalization poor. The depth of an ANN is the number of hidden layers in an ANN. It is the increase of computational power in the time of computers that have made deep ANNs possible and therefore also advancements in the fields of e.g. image processing and speech recognition [20].

Convolutional Neural Network

In DL there are different architectures. One of the most common DL architecture for image processing is called convolutional neural network (CNN). In general, a CNN typically takes an image as input, this is then passed through multiple convolutional and pooling layers before entering a part of fully connected layers which produce the final output [21]. Flattening, the process of converting a matrix into a column vector, is also needed before the data is fed to the fully connected layers. Figure 2.2 illustrates these main components of a CNN. Minimal pre-processing is needed with CNNs because a CNN takes raw images as input; however, a CNN requires fixed-size images [4].

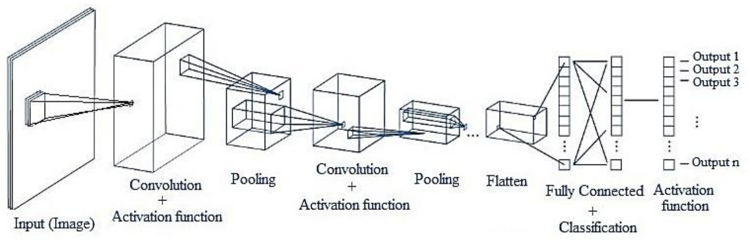

Figure 2.2: An overview of a CNN. Creator: M. Hacibeyoglu [22]

Throughout the learning process of a CNN, the convolutional layers detects meaningful shapes like e.g. edges and circles in the input data. In the first convolutional layers basic shapes are detected compared to the later convolutional layers which detect more sophisticated shapes like for instance a face.

In the pooling layers which usually follow convolutional layers, the purpose is to downsize the spatial data representation and to decrease the risk of overfitting. A pooling layer uses a pooling operation on the input data, a commonly called average pooling. Average pooling calculates the average of the numbers in the filter, which is slid across the input data, and passes it to the pooled feature map as illustrated in figure 2.3.

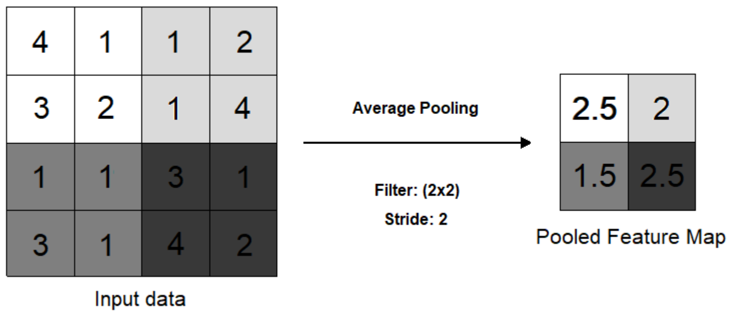

Figure 2.3: An example showing the result of the average pooling operation.

In the end of a CNN there is a part with a fully connected ANN. This ANN will combine feature produced by the earlier part of a CNN in order to classify the input image [20]. Some CNNs have more than one fully connected ANN at the end of the CNN, one example of this is the AlexNet [23].

2.4 Transfer Learning

Transfer learning is, as the name suggests, a way to transfer knowledge learned in one or more source tasks and use it to improve learning in a related task. By doing this, an already trained network can be used to solve new similar problems in a more efficient and quicker way. This is in contrast to traditional ML, which is designed to learn each task from scratch and in each of these models, brand new training data is required [24].

The more the source task and the target task are related, the better the prediction of the newly trained transfer network will be. A major challenge in the development of transfer methods is to avoid producing negative transfer (meaning that the transfer method decreases performance, compared to performance without transfer) between tasks that are less related.

There are several approaches to take in order to avoid negative transfer. One way is to try to recognize and reject harmful source-task information while learning the target task, in order to minimize the impact of bad information so that the performance of the transfer is as least as good as the performance would be without transfer.

If there not only exists one source task, but instead a set of candidate tasks, then one could try to choose the "best" source task out of the candidates in order to avoid negative transfer. There are multiple factors one could consider when choosing how "best" in this situation is defined. For example, the source tasks could be ordered by their difficulty. In that case, appropriate source tasks would be those that are less difficult than the target task, but still contain useful information. Another example for finding the best source tasks is to construct a graph out of the source and target tasks, and then find the most similar tasks by looking for graph isomorphism [25].

With transfer learning, only the fully connected layers of the source model need to be retrained in order to create the target model [24].

2.5 Performance Metrics

In order to compare different DL models for DR grading classification, different performance metrics are needed. Below are the most commonly used metrics in research regarding DR and ML [4].

Confusion Matrix

A confusion matrix is used to visualize the outcome of a classifier [26]. When using ML, the learning has to be supervised in order to produce a confusion matrix because only then there is a set number of labels which make the classes in a confusion matrix. This is not necessarily a metric and it is often not shown in similar research, although we show it to help the reader get familiar with the fundamental concepts.

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	**5**	2	2	1	3
	1	1	**7**	1	3	4
	2	4	2	**7**	0	1
	3	2	3	1	**6**	3
	4	0	1	2	3	**7**

Table 2.1: An example of a confusion matrix used for multiclass classification. The bold values in the diagonal of the confusion matrix are the desired outcomes.

There are two sorts of classifications: binary and multiclass. In binary classification there exist two classes; thus, the confusion matrix will have a dimension of 2x2. In multiclass classification, three or more classes exist and then the confusion matrix has equally many rows and columns as the number of classes.

In binary classification, a desired (true) outcome occurs when a classifier predicts the same class equal to the actual class and all desired outcomes are represented as values in the diagonal of a confusion matrix. A desired outcome is either called *true positive* ($\mathcal{TP}$) or *true negative* ($\mathcal{TN}$). If only class 0 (positive) and 1 (negative) was present in figure 2.1, it would be binary classification with $\mathcal{TP} = 5$ and $\mathcal{TN} = 7$. A false outcome occurs when the classifier predicts an unqualified class compared to the actual class. These outcomes are either called *false positive* ($\mathcal{FP}$) or *false negative* ($\mathcal{FN}$), in figure 2.1 these would become $\mathcal{FP} = 2$ and $\mathcal{FN} = 1$.

This can be generalized to multiclass classification which this study employs because of the five labels (or classes) in IDRiD (the dataset used in this study, see section 3.1). Table 2.1 and 2.2 are based on the labels from IDRiD. In multiclass classification, there are multiple separate $\mathcal{TP}$; one for each class along the diagonal of a confusion matrix. The $\mathcal{TP}$ value for class 0, 1, 2, 3 and 4 in table 2.1 and is $\mathcal{TP}_0 = 5$, $\mathcal{TP}_1 = 7$, $\mathcal{TP}_2 = 7$, $\mathcal{TP}_3 = 6$ and $\mathcal{TP}_4 = 7$, respectively. There are also multiple separate $\mathcal{TN}$, one per class which is calculated by taking the sum of all entities in a confusion matrix excluding those in the row and column belonging to the class which $\mathcal{TN}$ is calculated for. In table 2.2, all the partial $\mathcal{TN}$ are shown in order to calculate the $\mathcal{TN}$ for class 1. Calculating $\mathcal{FP}$ for a class is performed by adding all the entities in the row of the class, excluding its $\mathcal{TP}$, in a confusion matrix. To calculate $\mathcal{FN}$, all the entities in the column of a specific class are summed up excluding the $\mathcal{TP}$ of the class. For class 1 in table 2.2, $\mathcal{FP}_1 = E_{10} + E_{12} + E_{13} + E_{14}$ and $\mathcal{FN}_1 = E_{01} + E_{21} + E_{31} + E_{41}$. Based on this, the following metrics can be calculated [26].

		Actual Class				
		0	*1*	2	3	4
Predicted Class	*0*	$\mathcal{TN}_{00}$	E_{01}	$\mathcal{TN}_{02}$	$\mathcal{TN}_{03}$	$\mathcal{TN}_{04}$
	1	E_{10}	$\mathcal{TP}_1$	E_{12}	E_{13}	E_{14}
	2	$\mathcal{TN}_{20}$	E_{21}	$\mathcal{TN}_{22}$	$\mathcal{TN}_{23}$	$\mathcal{TN}_{23}$
	3	$\mathcal{TN}_{30}$	E_{31}	$\mathcal{TN}_{32}$	$\mathcal{TN}_{33}$	$\mathcal{TN}_{34}$
	4	$\mathcal{TN}_{40}$	E_{41}	$\mathcal{TN}_{42}$	$\mathcal{TN}_{43}$	$\mathcal{TN}_{44}$

Table 2.2: An example of the different symbols required to calculate all four outcomes of class 1 in multiclass classification.

Accuracy

The ratio between the desired outcomes and the total number of entities. The accuracy of a classifier is calculated by dividing the sum of the diagonal values by the sum of all the entities in its corresponding confusion matrix.

Recall (Sensitivity)

In multiclass classification, the specificity is calculated per class. The recall per class is calculated, from a confusion matrix, by dividing the desired outcome value (diagonal value) of a class by the sum of all the entities in the actual class column. With the symbols presented above $Recall_{class} = \mathcal{TP}_{class}/(\mathcal{TP}_{class} + \mathcal{FN}_{class})$.

Specificity

As for recall, specificity is calculated per class in multiclass classification. The specificity per class is calculated as $specificity_{class} = \mathcal{TN}_{class}/(\mathcal{TN}_{class} + \mathcal{FP}_{class})$.

Standard Deviation

A common metric used in statistics is sample standard deviation, denoted s. This metric shows the dispersion (or spread) of a set of values X from the mean (average) $\bar{X}$. It is calculated according to 2.1 where n is the sample size [27].

$$s = \sqrt{\frac{\sum (X - \bar{X})^2}{n - 1}} \tag{2.1}$$

2.6 Previous Work

This section will present previous work on DR grading classification with CNNs and inform about the state-of-the-art.

Carson et al. [7] studied, in 2018, the performance of a convolutional neural network on color fundus images for the recognition task of DR staging.

Two fundoscope image datasets were used for training. The first dataset used was a Kaggle dataset consisting of 35 000 images with 5-class labels (normal, mild, moderate, severe, end stage). Second dataset used was the Messidor-1 dataset, which consists of 1200 color fundus images with 4-class labels (normal, mild, moderate, severe).

They also used transfer learning with the pre-trained networks from the ImageNet database, which is a visual object recognition database. The pre-trained GoogLeNet model was trained on the Messidor dataset for 30 epochs using stochastic gradient descent optimization with step decay learning rate initialized at 0.002. The model validation achieved 66.03% as the best accuracy.

They trained GoogLeNet, AlexNet and VGG16 on binary-labeled (normal or mild vs moderate to end stage) Kaggle dataset, and found that the GoogLeNet achieved highest recall of 95% and specificity of 96%, and therefore achieved state-of-the-art accuracy. Their implementation was performed in Tensorflow and all model weights were allowed to be updated. However, they found after experiments with 3-ary and 4-ary classifiers with a GoogLeNet model on the Kaggle dataset that the recall levels for the mild class was much lower compared to recall levels for no DR and severe DR (7% for mild class, 98% and 93% for no DR and severe DR, respectively). Thus, they achieved state-of-the-art performance with CNNs using binary classifiers, but found that performance degrades with increasing number of classes. They conclude that the reason for this is that moderate and severe diabetic retinal images contain macroscopic features at a scale that current CNN architectures, such as those available from the ImageNet visual database, are optimized to classify. Conversely, the features that distinguish mild vs normal disease reside in less than 1% of the total pixel volume, a level of subtleness that is often difficult for human interpreters to detect.

They used a histogram equalization filtering algorithm as part of the image pre-processing and discovered that 3-ary classifier recall for the mild case increased from 0 to 29.4%, while this measure was approximately the same for the remaining two classes. The algorithm enabled improved detection of pinpoint subtle features and microaneurysms via convolutional filters, which were previously imperceptible by the CNN.

Pratt et al. [2] trained, in 2016, a CNN for classification of DR in fundus images. They used the Kaggle dataset consisting 80 000 images. With their proposed CNN, they achieved a recall of 95% and accuracy of 75% on 5000 validation images.

They used the DL package Keras with the Tearas ML backend. As part of pre-processing, they used colour normalization on all images using the OpenCV package. They also resized all images to 512x512 pixels in order to decrease memory size. They ran the CNN on the a high-end GPU, NVIDIA K40c, which contains 2880 CUDA cores and comes with the NVIDIA CUDA Deep Neural Network library (cuDNN) for GPU learning. However, their network encountered problems with distinguishing between mild, moderate and severe cases of DR. They write that "the low sensitivity, mainly from the mild and moderate classes, suggests the network struggled to learn deep enough features to detect some of the more intricate aspects of DR". Another issue they identified was that 10% of the images in the dataset they used were ungradable. They indicate that this could have been a significant hindering factor in the results since the images were missclassified for both training and validation.

Gulshan et. al. [28] trained, in 2016, a CNN architecture Inception-v3 to detect referable DR (defined as moderate and worse DR), referable diabetic macular edema, or both, on fundus images. They used two datasets; EyePACS-1 dataset, which consists of 9963 images taken from 4997 patients and MESSIDOR-2, which contains 1748 images. Both of these datasets were graded by at least seven US licensed ophthalmologists and ophthalmology senior residents. In two validation sets of 9963 images and 1748 images, at the operating point selected for high specificity, the algorithm had 90.3% and 87.0% recall and 98.1% and 98.5% specificity, respectively. At the operating point selected for high recall, the algorithm had 97.5% and 96.1% recall and 93.4% and 93.9% specificity, respectively.

From this brief literature review it can be seen that there was no uniform database which the different CNNs were trained and validated on, supported by [4]. This, in addition to differently used performance metrics, makes comparison hard to determine a state-of-the-art. Although, it seems like the CNN by Gulshan et al. [28] outperformed the others and therefore it makes the current state-of-the-art as concluded by authors themselves.

Chapter 3

Methods

This chapter presents chosen methods to carry out the study. First, relevant information about the used dataset is explained. Then, a section about the pre-trained network used in the study follows. After this, details on how the implementation was performed follows, such as data pre-processing, how the CNN was built, how the fully connected layers were trained and finally how the network was fine-tuned. Lastly in this chapter, a section about how the network was evaluated follows.

3.1 Dataset

In this study, the Indian Diabetic Retinopathy Image Dataset (IDRiD) was used to train and test the CNN. It is divided into three parts: segmentation, disease grading and localization. The part used in this study was disease grading which consists of 516 labeled fundus images (where 413 and 103 images are meant for training and testing, respectively). Each image has a corresponding label value between 0 and 4 which represents the disease severity. The values has the following meanings: no DR (0), mild DR (1), moderate DR (2), severe DR (3) and proliferative DR (4). These labels are found in supplied csv files (also divided into training and testing).

The dataset was publicly published 2018 and the images were acquired from an eye clinic in the city Nanded located in India. The images were taken during the time period 2009 to 2017. In 2018, the dataset was part of a challenge with the aim of evaluating automated DR grading classification algorithms [29].

3.2 Pre-trained Network

The CNN that is used as a pre-trained network, thereby obtaining the utility of transfer learning, is called Inception-v3 [30]. Inception-v3 is the third version Inception CNN developed at Google which is more compatible to scale up in some applications without losing excessive performance compared to the prior versions. Other CNNs like the AlexNet [23] requires more memory and computational resources than an Inception CNN. Inception-v3 (and all the previous versions) was trained on the ImageNet dataset [31] which originally had 3.2 million images.

The Inception-v3 was used because it has achieved good results in previous studies like [28] and due to the fact that less resources were needed.

3.3 Implementation

A software library called TensorFlow and the programming language Python were used to implement the CNN. TensorFlow was used due to its simplicity when trying to build and train a CNN and because it has been used in a previous study [7]. Furthermore, Keras, which is a neural network library running on top of TensorFlow, has the Inception-v3 available for usage as a pre-trained network.

The hardware used to build and train the CNN was a laptop with a Intel Core i7-7500U CPU and 8 GB of RAM. This hardware was also used for evaluation (section 3.4). No dedicated GPU was installed on the laptop and therefore the CPU was used instead for training which is generally not considered optimal [32].

The following subsections are presented in the same order as they were implemented.

3.3.1 Data Pre-processing

In the data pre-processing stage, the images from IDRiD were first cropped to a size of 1200x1200 pixels. This was done to remove most of the black pixels outside the circular border in the images, see figure 3.1. These black pixels are irrelevant and do not provide any useful information for the CNN. Without the black pixels we got better results.

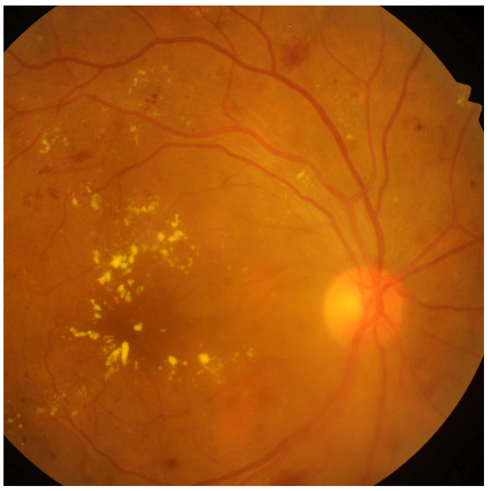

Figure 3.1: The cropped version of the fundus image shown in figure 2.1.

The images were then resized from 1200x1200 pixels to a size of 125x125 pixels. Resizing was necessary to perform due to limited hardware.

After resizing, a histogram equalization algorithm was applied to the images. This was used for image contrast enhancement and we used the histogram equalization algorithm implemented in the OpenCV library. The improvement Carson et al. [7] obtained when using the algorithm in pre-processing, where it lead to an increase of 3-ary classifier recall for the mild case increased from 0 to 29.4%, was the motivation for using the histogram equalization algorithm in our study.

Lastly, all the images were normalized, a process of changing the range of all the pixel values in an image. In our case, all the images were divided by 255 (the highest pixel value in RGB images) resulting in pixel values between 0 and 1. Normalization was performed to make the images on the same scale [33].

3.3.2 Building the CNN

The Inception-v3 CNN was imported from Keras with the option of not including the fully connected layers, as these layers are not applicable to DR grading classification, and with the input shape set to 125x125x3 (same as the pre-processed images). The value three in the input shape represents the different color channels (e.g. red, green and blue when using RGB). The weights coming from the training of the Inception-v3 on the ImageNet datset were also included in this step. Transfer learning is utilized by importing and using the

Inception-v3 CNN as the base of the constructed CNN model described in this subsection.

When passing the pre-processed images latter to this part of the constructed CNN, it will produce a block of features per image with the shape 2x2x2048. The weights in this part were set to be non-trainable because when training the CNN first-time, only the weights in the fully connected layers should be trained. This is sometimes referred to as freezing the layers.

A pooling layer was subsequently added to the CNN serving as a flattening layer. The pooling layer performed average pooling, having no stride and a filter size of 2x2, on the block of features. This transformed the block of features into a vector of 2048 elements. It was done in order to pass the data to the fully connected layers which require a vector as input. The fully connected layers were then joined together with the flattening layer. Its last (output) layer was set to have five nodes, one for each label occurring in IDRiD. Softmax was used as the activation function in the fully connected layer because it functions well when working with multiclass classification [34].

3.3.3 Training the Fully Connected Layers

Before the actual training was started, it was necessary to specify an optimizer, loss function and metric. The loss function used, as it is appropriate for multiclass classification, was sparse categorical crossentropy from Keras built-in loss functions. The optimizer used was RMSprop and its learning rate was set to 0.0001 (default is 0.001) because this produced better results. In addition to the loss function, accuracy was chosen as the metric in order to show how the CNN model progressed and performed during training. Training was subsequently performed on the training images from IDRiD. The CNN output was compared with its input image corresponding label obtained from the csv training file in order to learn. As mentioned above, this training was only meant for the fully connected layers. From the training images, 10% was taken for usage as a validation set. Images in the validation set are not used for training the CNN, instead they are used to check if the CNN model is overfitting on the training images. The training was performed for 10 epochs where one epoch means that the CNN model goes through all of the input images once. Different epoch values was tested although 10 gave acceptable results while avoiding excessive overfitting.

The CNN was trained (including fine-tuning, see subsection 3.3.4) and evaluated (see subsection 3.4) twenty times in order to see how it performed on average. This was done because each CNN model performed differently after each training session on the images from IDRiD. The specific value of twenty was chosen because it seemed enough to obtain a general idea of how the CNN model performed on average while having limited time.

With the hardware described earlier, it took approximately 1 minute, on average of all training sessions, to train and validate the CNN for 10 epochs on the training images from IDRiD (371 and 42 images were used for training and validation, respectively).

3.3.4 Fine-tuning

Fine-tuning is the next optional step in the process of completing a CNN model, used to increase the performance of the CNN model [32]. When fine-tuning a CNN model, all the weights throughout a CNN are set to be trainable (or a great portion of them) and not only the weights in the fully connected layers as before.

All the weights were set to be trainable in the built and trained CNN models mentioned in the previous subsections. The same optimizer (including learning rate), loss function and metric were used as before. With the same configurations as previously, the CNN models were further trained for 10 final epochs.

It took approximately 6 minutes, on average of all training sessions, to train the CNN models. This was performed with the same epoch value which resulted in a significant increase of time compared to when only training the fully connected layers. It is evident that training the entire CNN is more time consuming.

3.4 Evaluation

In order to compare all the twenty trained CNN models with the results from previous studies in this field, it was needed to measure the performance of the models. Both the training and testing images were evaluated although the performance of each model on the testing images have a greater importance. The testing images were not used in the training of the models and therefore the models had not seen these images before. A good model performance on the testing images could indicate an equally good performance on other new fundus images and thus give an idea of if the model could be used in practice.

All the images were passed to the trained CNN models in order to obtain a predicted label. Each predicted label and the correct label were recorded separately for the training and testing images resulting in two confusion matrices (see table 4.3 (a) & (b)) for each model. From these confusion matrices, the accuracy, recall and specificity of each CNN model were calculated according to section 2.5.

Lastly, the accuracy, recall and specificity of each model, in the different training sessions, were collected. This was done to calculate the average accuracy, recall, specificity and the standard deviation as seen in table 4.4, 4.5 (a) & (b) and 4.6 (a) & (b).

Chapter 4

Results

This section will present the results achieved during the training and evaluation of the implemented CNN models mentioned in the Method section (chapter 3). The CNN models, in all twenty training sessions, were based on the pre-trained Inception-v3 network and trained on images from IDRiD for 20 epochs in total. The aim in this report is to investigate the performance, using accuracy, recall and specificity, of a CNN for DR grading classification in fundus images with the use of transfer learning. Transfer learning is utilized by using a pre-trained network.

4.1 Performance and Outcome

All the results shown in this subsection comes from the evaluation step (section 3.4). Hence, the measurements are split into training and testing.

Twenty training sessions were performed and due to repetitive data, only the results from one training session are shown. Table 4.1, 4.2 (a) & (b) and 4.3 (a) & (b) show the results of training session 1. See Appendix A for a complete record of results from all training sessions.

In training session 1 the CNN model achieved an accuracy of 77.0% and 36.9%, as shown in table 4.1, on the training and testing images, respectively. The accuracies of DR grading on both the training and testing images were lower in this session compared to the average accuracy (see table 4.4) of all twenty training sessions.

	Accuracy (%)
Training images	77.0
Testing images	36.9

Table 4.1: The accuracy in percentage of the CNN model for DR grading classification on the training and testing images from IDRiD. This accuracy was obtained from training session 1.

In table 4.2 (a) & (b), the recall and specificity per class are shown from DR grading classification on the training and testing images, obtained from training session 1. The 0.0% recall and 100.0% specificity for class 1 in both table 4.2 (a) & (b) is in accordance with the average recall and specificity in table 4.5 (a) & (b).

Class	**Recall (%)**	**Specificity (%)**
0	94.8	89.6
1	0.0	100.0
2	83.1	97.5
3	98.6	82.6
4	10.2	100.0

(a) Training images

Class	**Recall (%)**	**Specificity (%)**
0	52.9	82.6
1	0.0	100.0
2	21.9	93.0
3	68.4	42.9
4	0.0	100.0

(b) Testing images

Table 4.2: The recall and specificity in percentage per class of the CNN model for DR grading classification on the training and testing images from IDRiD. These result values were obtained from training session 1.

The CNN model in training session 1 never predicted class 1 as seen in table 4.3 (a) & (b). This resulted in the low recall and high specificity of classification of class 1 in table 4.2 (a) & (b). This occurred in all training sessions as seen in table 4.5 (a) & (b). The recall and specificity have a standard deviation of 0.0% for class 1, see table 4.6 (a) & (b). If adding all the entities in each column of both table 4.2 (a) & (b), class 1 has the lowest sum which means that IDRiD has few images graded as mild DR.

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	**127**	15	9	1	4
	1	0	**0**	0	0	0
	2	2	2	**113**	0	3
	3	5	3	14	**73**	37
	4	0	0	0	0	**5**

(a) Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	**18**	2	5	3	2
	1	0	**0**	0	0	0
	2	2	0	**7**	3	0
	3	14	3	20	**13**	11
	4	0	0	0	0	**0**

(b) Testing images

Table 4.3: Confusion matrices showing the outcome of the CNN model when classifying the grading of DR on the training and testing images from IDRiD. It shows the outcome of training session 1.

Table 4.4 shows the average accuracy of all training sessions in percentage for DR grading classification on both the training and testing images from IDRiD together with the respective standard deviation. It can be seen that on average the accuracy for DR grading is higher when classifying on the training images compared to the testing images. There is no substantial difference between the standard deviation for the training and testing images.

	Average accuracy (%)	**Standard deviation (%)**
Training images	79.2	3.5
Testing images	42.8	3.8

Table 4.4: The average accuracy of all twenty training sessions for the training and testing images together with the respective standard deviations in percentage.

Table 4.5 (a) & (b) shows the average recall and specificity per class when performing DR grading on the training and testing images from IDRiD. Class 1 has both the lowest average recall and highest average specificity in both table 4.5 (a) & (b). On the training images shown in table 4.5 (a), the average recall is 94.1% for class 0 which is the highest average recall out of all the classes. This is in accordance with the testing images in table 4.5 (b) where class 0 also has the highest average recall (59.7%). The lowest average specificity occurred in class 2 on both the training and testing images (86.1% and 62.4%, respectively). Table 4.5 (b) shows a low average recall of 0.8% for class 4 which is significantly lower than the average recall for class 4 in table 4.5 (a).

Class	Average recall (%)	Average specificity (%)
0	94.1	91.3
1	0.0	100.0
2	93.0	86.1
3	87.9	93.0
4	19.6	96.6

(a) Training images

Class	Average recall (%)	Average specificity (%)
0	59.7	77.8
1	0.0	100.0
2	51.0	62.4
3	38.7	80.2
4	0.8	99.7

(b) Testing images

Table 4.5: The average recall and specificity per class of all twenty training sessions for the training and testing images displayed in percentage. Table 4.6 shows the associated standard deviations.

Table 4.6 (a) & (b) show the standard deviation of the recall and specificity for each class in percentage. These standard deviations are used together with the average recalls and specificties in table 4.5 (a) & (b) to show the amount of spread of the recall and specificity per class in the twenty training sessions. Overall, the standard deviations of the recalls and specificities are higher in table 4.6 (b) than in table 4.6 (a).

Class	Standard deviation of recall (%)	Standard deviation of specificity (%)
0	3.5	3.2
1	0.0	0.0
2	3.8	6.3
3	6.8	4.4
4	23.0	15.0

(a) Training images

Class	Standard deviation of recall (%)	Standard deviation of specificity (%)
0	15.0	7.1
1	0.0	0.0
2	14.4	14.0
3	18.2	12.7
4	3.4	0.6

(b) Testing images

Table 4.6: The standard deviations of the recall and specificity per class for the training and testing images in percentage. Table 4.5 shows the associated average recalls and specificities.

4.2 CNN Model Progression

The learning progressions of our CNN models in the different training sessions are relevant in order to see how the models perform throughout the training in the different epochs. The CNN model progression in training session 1 is shown in figure 4.1 which displays both the accuracy of the model and loss when classifying DR grading on the training and validation images. We only show the progression of a model from one training session because all twenty model progressions look similar.

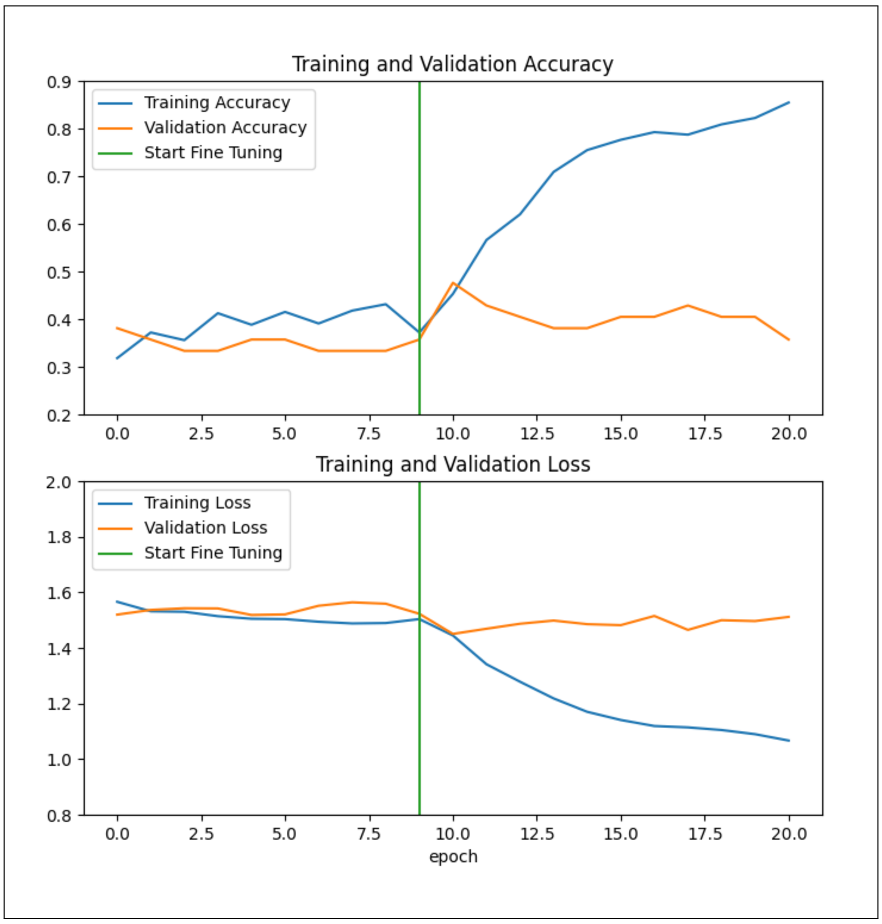

Figure 4.1: The upper plot shows the accuracy of the CNN model on the training and validation images after each epoch. The lower plot shows the loss on the training and validation images after each epoch. In both plots, the start of the fine-tuning step is displayed as a green line. This is the progression of the CNN model in training session 1.

Figure 4.1 (upper plot) shows a significant increase of the model accuracy on the training and testing images after the fine-tuning step. On the validation images, not used in the actual training, the model accuracy starts to decrease after 1 epoch after the fine-tuning step.

There is almost no change in loss before the fine-tuning step as seen in the lower plot of figure 4.1. After the fine-tuning step, the training loss started to decrease while the validation loss first decreased for 1 epoch to then flatten out.

It should be pointed out that the accuracy of the CNN model in training session 1, after 20 epochs, on the training images is unequal to the accuracy of the same training session model on the training images presented in table 4.1. This is because 10% of the training images from IDRiD were taken for validation. The model performs better on the training images than the validation images as seen in figure 4.1. When including the validation images as the training images in the evaluation step, the model accuracy on the training images was reduced.

Chapter 5

Discussion

The performance of the CNN models, in the different training sessions, achieved lower results than state-of-the-art results. An average testing accuracy of 42.8% was achieved with a standard deviation of 3.8%. There are a few factors that we think are likely to be the reasons to why the results are lower than state-of-the-art results. One of these factors is that the CNN in each training sessions was trained and fine-tuned on a relatively small amount of images (516). If it would have been trained on a much larger dataset, the performance would likely be higher. Data augmentation is a common way to deal with too small training sets. Both [2] and [7] used real-time data augmentation to improve the localization ability of the network.

Another factor is that the results could possibly have improved if more steps would have been added as part of pre-processing. By more pre-processing of images and by using real-time data augmentation, the performance might would have increased.

Another aspect is the cropping and resizing steps of images which were performed as part of pre-processing, due to performance reasons. It is possible that some features were removed or neglected for this reason, and thus not being considered by the CNN models during training and testing. This could possibly have lead to a misclassification of images or that the CNN did not make a classification at all. This is a relevant aspect to bring up when discussing factors affecting the results, since the images were first cropped and then significantly resized, from 1200x1200 pixels to 125x125 pixels, which likely could have affected the results in some way.

Also, it is relevant to address that the CNN-models used in the studies mentioned in previous work were trained on different datasets than this study. Most of the datasets used were also of very large volume, which require more powerful hardware than possible for this study. Thus, by having access to more powerful hardware to implement and train the CNN models with datasets of larger volume, the results could have possibly increased.

Another factor which possibly could have had an impact on the results, is the imbalance between representation of classes in the used dataset. From training session 1, table 4.3 (a), it can be seen that class 1 (mild DR) is significantly less occurring in the dataset compared to other classes. Specifically, of the training images, there are 134 images belonging to class 0, 20 images of class 1, 136 of class 2, 74 of class 3 and 49 of class 4. This means that the CNN model got less training on some classes compared to others, class 1 being the least trained on. This is in accordance with our results from training sessions, in which class 1 never got predicted, as seen in table 4.3 (a) & (b). This occurred in all training session resulting in the $0.0 \pm 0.0\%$ average recall and $100.0 \pm 0.0\%$ average specificity for class 1 shown in table 4.5 (a) & (b) combined with table 4.6 (a) & (b). Furthermore, the second least occurring class in the dataset was class 4 (proliferative DR). Class 4 had a testing average of 97.7% for specificity and 0.8% for recall, as seen in table 4.5 (a). The low recall is in line width the relatively low occurrence of images belonging to this class.

The low occurrences of images of grade 1 and grade 4 is not acceptable because it is wanted for the CNN models to classify all DR grades with a high recall and specificity as possible. The reason behind the fact that class 1 was the lowest occurring class could be that it is hard to detect the early signs of DR. If calculating the sum of all the entities in each column of table 4.3 (a) % (b), column 0 and 2 have the highest sum compared to all the other classes and could indicate that images which should have been labeled as 1 was instead labeled either 0 or 2. A recall of 7% for mild DR was achieved by Carson et al. [7] compared to 0.0% in this study and together with Pratt et al. [2], all experience the same problem of a low recall for mild DR. This is also despite the fact that different datasets were used which means that other DR grading datasets possibly also contain few images labeled as 1. To remedy this potential problem, more images labeled as mild DR has to be included in DR datasets or better labeling has to take place.

Chapter 6

Conclusion

The aim of the study was to investigate the following question:

> *What is the performance of a convolutional neural network for diabetic retinopathy grading classification in fundus images with the use of transfer learning?*

In this study, we have implemented a CNN using transfer learning, meaning that we have used a pre-trained CNN model (Inception-v3), which we then further trained and fine-tuned on our research area in order to perform DR grading classification on fundus images. Twenty training sessions were performed which resulted in testing accuracies in the range of 35% to 48.5%. The average testing recall achieved for class 0, 1, 2, 3 and 4 was 59.7%, 0.0%, 51.0%, 38.7% and 0.8%, respectively. The average testing specificity achieved for class 0, 1, 2, 3 and 4 was 77.8%, 100.0%, 62.4%, 80.2% and 99.7%, respectively. Lower results were achieved in this study compared to the previous work which we think is partially a consequence of a small size dataset, as well as an imbalance of occurrence of classes in the used dataset. Our results clearly show the need for better labeling in DR grading datasets to make each class equally represented, left for future research.

Bibliography

[1] World Health Organization. *Global report on diabetes: executive summary*. Technical documents. 2016.

[2] Harry Pratt et al. "Convolutional neural networks for diabetic retinopathy". In: *Procedia Computer Science* 90 (2016), pp. 200–205.

[3] Rajiv Raman et al. "Diabetic retinopathy: An epidemic at home and around the world". In: *Indian journal of ophthalmology* 64.1 (2016), p. 69.

[4] Norah Asiri et al. "Deep learning based computer-aided diagnosis systems for diabetic retinopathy: A survey". In: *Artificial Intelligence in Medicine* 99 (2019), p. 101701. ISSN: 0933-3657. DOI: https://doi.org/10.1016/j.artmed.2019.07.009. URL: http://www.sciencedirect.com/science/article/pii/S0933365718307607.

[5] Muthu Rama Krishnan Mookiah et al. "Computer-aided diagnosis of diabetic retinopathy: A review". In: *Computers in Biology and Medicine* 43.12 (2013), pp. 2136–2155. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2013.10.007. URL: http://www.sciencedirect.com/science/article/pii/S0010482513002862.

[6] Robyn J. Tapp et al. "The Prevalence of and Factors Associated With Diabetic Retinopathy in the Australian Population". In: *Diabetes Care* 26.6 (2003), pp. 1731–1737. ISSN: 0149-5992. DOI: 10.2337/diacare.26.6.1731. eprint: https://care.diabetesjournals.org/content/26/6/1731.full.pdf. URL: https://care.diabetesjournals.org/content/26/6/1731.

[7] Carson Lam et al. "Automated detection of diabetic retinopathy using deep learning". In: *AMIA Summits on Translational Science Proceedings* 2018 (2018), p. 147.

[8] Wynne Hsu et al. “The role of domain knowledge in the detection of retinal hard exudates”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 2. IEEE. 2001, pp. II–II.

[9] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. “Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation”. In: *AI 2006: Advances in Artificial Intelligence*. Ed. by Abdul Sattar and Byeong-ho Kang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1015–1021. ISBN: 978-3-540-49788-2.

[10] Mohammad Abdullah-Al-Wadud et al. “A dynamic histogram equalization for image contrast enhancement”. In: *IEEE Transactions on Consumer Electronics* 53.2 (2007), pp. 593–600.

[11] Alan W Stitt et al. “The progress in understanding and treatment of diabetic retinopathy”. In: *Progress in retinal and eye research* 51 (2016), pp. 156–186.

[12] CP Wilkinson et al. “Proposed international clinical diabetic retinopathy and diabetic macular edema disease severity scales”. In: *Ophthalmology* 110.9 (2003), pp. 1677–1682.

[13] Gwenolé Quellec et al. “Deep image mining for diabetic retinopathy screening”. In: *Medical image analysis* 39 (2017), pp. 178–193.

[14] Michael D Abràmoff, Mona K Garvin, and Milan Sonka. “Retinal imaging and image analysis”. In: *IEEE reviews in biomedical engineering* 3 (2010), pp. 169–208.

[15] George H Bresnick et al. “A screening approach to the surveillance of patients with diabetes for the presence of vision-threatening retinopathy”. In: *Ophthalmology* 107.1 (2000), pp. 19–24.

[16] JL Kinyoun et al. “Ophthalmoscopy versus fundus photographs for detecting and grading diabetic retinopathy.” In: *Investigative ophthalmology & visual science* 33.6 (1992), pp. 1888–1893.

[17] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[18] Igor Kononenko. “Machine learning for medical diagnosis: history, state of the art and perspective”. In: *Artificial Intelligence in medicine* 23.1 (2001), pp. 89–109.

[19] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. “Supervised machine learning: A review of classification techniques”. In: *Emerging artificial intelligence applications in computer engineering* 160 (2007), pp. 3–24.

[20] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner’s Approach*. 1st. O’Reilly Media, Inc., 2017. ISBN: 1491914254.

[21] Waseem Rawat and Zenghui Wang. “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review”. In: *Neural Computation* 29.9 (2017). PMID: 28599112, pp. 2352–2449. DOI: `10.1162/neco\_a\_00990`. eprint: `https://doi.org/10.1162/neco_a_00990`. URL: `https://doi.org/10.1162/neco_a_00990`.

[22] Mehmet Hacibeyoglu. “Human Gender Prediction on Facial Mobil Images using Convolutional Neural Networks”. In: *International Journal of Intelligent Systems and Applications in Engineering* 3 (Sept. 2018), pp. 203–208. DOI: `10.18201/ijisae.2018644778`.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[24] Jie Lu et al. “Transfer learning using computational intelligence: A survey”. In: *Knowledge-Based Systems* 80 (2015), pp. 14–23.

[25] Emilio Soria Olivas. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*. IGI Global, 2009.

[26] Alaa Tharwat. “Classification assessment methods”. In: *Applied Computing and Informatics* (2018). ISSN: 2210-8327. DOI: `https://doi.org/10.1016/j.aci.2018.08.003`. URL: `http://www.sciencedirect.com/science/article/pii/S2210832718301546`.

[27] M. P. Barde and P. J. Barde. “What to use to express the variability of data: Standard deviation or standard error of mean?” In: *Perspectives in clinical research* 3(3) (2012), pp. 113–116. URL: `https://doi.org/10.4103/2229-3485.100662`.

[28] Varun Gulshan et al. “Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs”. In: *Jama* 316.22 (2016), pp. 2402–2410.

[29] Prasanna Porwal et al. “Indian Diabetic Retinopathy Image Dataset (IDRiD): A Database for Diabetic Retinopathy Screening Research”. In: *Data* 3.3 (2018). ISSN: 2306-5729. DOI: `10.3390/data3030025`. URL: `https://www.mdpi.com/2306-5729/3/3/25`.

[30] Christian Szegedy et al. “Rethinking the Inception Architecture for Computer Vision”. In: June 2016. DOI: `10.1109/CVPR.2016.308`.

[31] J. Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[32] N. Tajbakhsh et al. “Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?” In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1299–1312.

[33] R. Ghosh, K. Ghosh, and S. Maitra. “Automatic detection and classification of diabetic retinopathy stages using CNN”. In: *2017 4th International Conference on Signal Processing and Integrated Networks (SPIN)*. 2017, pp. 550–554.

[34] John S. Bridle. “Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition”. In: *Neurocomputing*. Ed. by Françoise Fogelman Soulié and Jeanny Hérault. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 227–236. ISBN: 978-3-642-76153-9.

Appendix A

Complete Record of Results from all Training Sessions

Session 1

	Accuracy (%)
Training images	77.0
Testing images	36.9

Class	Recall (%)	Specificity (%)
0	94.8	89.6
1	0.0	100.0
2	83.1	97.5
3	98.6	82.6
4	10.2	100.0

Training images

Class	Recall (%)	Specificity (%)
0	52.9	82.6
1	0.0	100.0
2	21.9	93.0
3	68.4	42.9
4	0.0	100.0

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	127	15	9	1	4
	1	0	0	0	0	0
	2	2	2	113	0	3
	3	5	3	14	73	37
	4	0	0	0	0	5

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	18	2	5	3	2
	1	0	0	0	0	0
	2	2	0	7	3	0
	3	14	3	20	13	11
	4	0	0	0	0	0

Testing images

Session 2

	Accuracy (%)
Training images	84.0
Testing images	47.6

Class	**Recall (%)**	**Specificity (%)**
0	95.0	93.5
1	0.0	100.0
2	93.4	88.4
3	93.2	95.6
4	46.9	99.7

Training images

Class	**Recall (%)**	**Specificity (%)**
0	64.7	73.9
1	0.0	100.0
2	46.9	69.0
3	63.1	84.5
4	0.0	98.9

Testing images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	128	10	3	0	5
1	0	0	0	0	0
2	6	6	127	5	15
3	0	4	5	69	6
4	0	0	1	0	23

Training images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	22	3	9	2	4
1	0	0	0	0	0
2	11	1	15	5	5
3	0	1	8	12	4
4	1	0	0	0	0

Testing images

Session 3

	Accuracy (%)
Training images	78.2
Testing images	42.7

Class	**Recall (%)**	**Specificity (%)**
0	96.3	90.3
1	0.0	100.0
2	93.4	86.3
3	87.8	92.3
4	4.1	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	67.6	75.4
1	0.0	100.0
2	43.8	57.7
3	36.8	85.7
4	0.0	100.0

Testing images

Actual Class

Predicted Class	*0*	*1*	*2*	*3*	*4*
0	129	11	6	1	8
1	0	0	0	0	0
2	4	6	127	8	20
3	1	3	3	65	19
4	0	0	0	0	2

Training images

Actual Class

Predicted Class	*0*	*1*	*2*	*3*	*4*
0	23	3	9	3	2
1	0	0	0	0	0
2	8	2	14	9	11
3	3	0	9	7	0
4	0	0	0	0	0

Testing images

Session 4

	Accuracy (%)
Training images	76.3
Testing images	48.5

Class	**Recall (%)**	**Specificity (%)**
0	96.3	87.1
1	0.0	100.0
2	93.4	80.9
3	79.7	97.3
4	0.0	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	79.4	69.6
1	0.0	100.0
2	56.3	63.4
3	26.3	92.9
4	0.0	100.0

Testing images

Actual Class

Predicted Class	*0*	*1*	*2*	*3*	*4*
0	129	15	7	6	8
1	0	0	0	0	0
2	5	5	127	9	34
3	0	0	2	59	7
4	0	0	0	0	0

Training images

Actual Class

Predicted Class	*0*	*1*	*2*	*3*	*4*
0	27	4	11	3	3
1	0	0	0	0	0
2	5	1	18	11	9
3	2	0	3	5	1
4	0	0	0	0	0

Testing images

Session 5

	Accuracy (%)
Training images	76.8
Testing images	43.7

Class	Recall (%)	Specificity (%)
0	93.3	93.0
1	0.0	100.0
2	90.4	85.9
3	93.2	88.2
4	0.0	100.0

Training images

Class	Recall (%)	Specificity (%)
0	55.9	81.2
1	0.0	100.0
2	40.6	74.6
3	68.4	67.9
4	0.0	100.0

Testing images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	125	11	3	1	2
1	0	0	0	0	0
2	4	7	123	4	24
3	5	2	10	69	23
4	0	0	0	0	0

Training images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	19	3	6	2	2
1	0	0	0	0	0
2	9	1	13	4	4
3	6	1	13	13	7
4	0	0	0	0	0

Testing images

Session 6

	Accuracy (%)
Training images	78.5
Testing images	44.7

Class	Recall (%)	Specificity (%)
0	97.0	90.7
1	0.0	100.0
2	93.4	90.6
3	90.5	89.1
4	0.0	100.0

Training images

Class	Recall (%)	Specificity (%)
0	58.8	76.8
1	0.0	100.0
2	46.9	76.1
3	57.9	71.4
4	0.0	100.0

Testing images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	130	15	5	2	4
1	0	0	0	0	0
2	3	4	127	5	14
3	1	1	4	67	31
4	0	0	0	0	0

Training images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	20	3	6	4	3
1	0	0	0	0	0
2	10	1	15	4	2
3	4	1	11	11	8
4	0	0	0	0	0

Testing images

Session 7

	Accuracy (%)
Training images	73.4
Testing images	35.0

Class	**Recall (%)**	**Specificity (%)**
0	82.8	98.2
1	0.0	100.0
2	94.9	76.2
3	82.4	88.5
4	4.1	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	23.5	91.3
1	0.0	100.0
2	56.3	57.7
3	52.6	63.1
4	0.0	100.0

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	111	3	2	0	0
	1	0	0	0	0	0
	2	18	16	129	13	19
	3	5	1	5	61	28
	4	0	0	0	0	2

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	8	1	2	2	1
	1	0	0	0	0	0
	2	15	3	18	7	5
	3	11	1	12	10	7
	4	0	0	0	0	0

Testing images

Session 8

	Accuracy (%)
Training images	78.5
Testing images	40.8

Class	**Recall (%)**	**Specificity (%)**
0	96.3	92.8
1	0.0	100.0
2	95.6	79.8
3	87.8	96.2
4	0.0	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	44.1	82.6
1	0.0	100.0
2	65.6	46.5
3	31.6	86.9
4	0.0	100.0

Testing images

	Actual Class				
Predicted Class	*0*	*1*	*2*	*3*	*4*
0	129	6	3	2	9
1	0	0	0	0	0
2	5	14	130	7	30
3	0	0	3	65	10
4	0	0	0	0	0

Training images

	Actual Class				
Predicted Class	*0*	*1*	*2*	*3*	*4*
0	15	2	7	2	1
1	0	0	0	0	0
2	15	3	21	11	9
3	4	0	4	6	3
4	0	0	0	0	0

Testing images

Session 9

	Accuracy (%)
Training images	77.2
Testing images	45.6

Class	**Recall (%)**	**Specificity (%)**
0	96.3	86.7
1	0.0	100.0
2	86.0	91.3
3	93.2	90.3
4	8.2	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	73.5	72.5
1	0.0	100.0
2	34.4	81.7
3	57.9	71.4
4	0.0	100.0

Testing images

	Actual Class				
Predicted Class	*0*	*1*	*2*	*3*	*4*
0	129	15	10	3	9
1	0	0	0	0	0
2	4	3	117	2	15
3	1	2	9	69	21
4	0	0	0	0	4

Training images

	Actual Class				
Predicted Class	*0*	*1*	*2*	*3*	*4*
0	25	3	11	3	2
1	0	0	0	0	0
2	4	2	11	5	2
3	5	0	10	11	9
4	0	0	0	0	0

Testing images

Session 10

	Accuracy (%)
Training images	81.4
Testing images	39.8

Class	**Recall (%)**	**Specificity (%)**
0	94.0	91.8
1	0.0	100.0
2	93.4	85.9
3	90.5	95.6
4	32.7	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	47.1	81.2
1	0.0	100.0
2	62.5	50.7
3	26.3	83.3
4	0.0	100.0

Testing images

Actual Class

Predicted Class	*0*	*1*	*2*	*3*	*4*
0	126	12	6	2	3
1	0	0	0	0	0
2	8	7	127	5	19
3	0	1	3	67	11
4	0	0	0	0	16

Training images

Actual Class

Predicted Class	*0*	*1*	*2*	*3*	*4*
0	16	3	5	2	3
1	0	0	0	0	0
2	13	2	20	12	8
3	5	0	7	5	2
4	0	0	0	0	0

Testing images

Session 11

	Accuracy (%)
Training images	77.0
Testing images	46.6

Class	**Recall (%)**	**Specificity (%)**
0	92.5	83.5
1	0.0	100.0
2	92.6	85.9
3	82.4	97.1
4	14.3	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	79.4	62.3
1	0.0	100.0
2	53.1	69.0
3	21.1	91.7
4	0.0	100.0

Testing images

Actual Class

Predicted Class	*0*	*1*	*2*	*3*	*4*
0	124	16	10	8	12
1	0	0	0	0	0
2	9	4	126	5	21
3	1	0	0	61	9
4	0	0	0	0	7

Training images

Actual Class

Predicted Class	*0*	*1*	*2*	*3*	*4*
0	27	3	11	7	5
1	0	0	0	0	0
2	6	2	17	8	6
3	1	0	4	4	2
4	0	0	0	0	0

Testing images

Session 12

	Accuracy (%)
Training images	86.9
Testing images	45.6

Class	**Recall (%)**	**Specificity (%)**
0	91.0	93.2
1	0.0	100.0
2	95.6	90.6
3	93.2	97.3
4	77.6	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	70.6	73.9
1	0.0	100.0
2	43.8	74.6
3	47.4	77.4
4	0.0	98.9

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	122	11	4	2	2
	1	0	0	0	0	0
	2	11	9	130	3	3
	3	1	0	2	69	6
	4	0	0	0	0	38

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	24	3	6	4	5
	1	0	0	0	0	0
	2	8	2	14	5	3
	3	2	0	12	9	5
	4	0	0	0	1	0

Testing images

Session 13

	Accuracy (%)
Training images	83.1
Testing images	43.7

Class	**Recall (%)**	**Specificity (%)**
0	98.5	92.5
1	0.0	100.0
2	86.8	94.9
3	94.6	90.0
4	46.9	99.7

Training images

Class	**Recall (%)**	**Specificity (%)**
0	76.5	78.3
1	0.0	100.0
2	31.3	71.8
3	47.4	75.0
4	0.0	97.8

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	132	13	6	1	1
	1	0	0	0	0	0
	2	1	5	118	3	5
	3	0	2	12	70	20
	4	1	0	0	0	23

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	26	2	8	3	2
	1	0	0	0	0	0
	2	7	2	10	5	6
	3	1	1	14	9	5
	4	0	0	0	2	0

Testing images

Session 14

	Accuracy (%)
Training images	75.8
Testing images	43.7

Class	**Recall (%)**	**Specificity (%)**
0	94.8	89.2
1	0.0	100.0
2	96.3	77.3
3	71.6	97.9
4	4.1	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	64.7	76.8
1	0.0	100.0
2	56.3	52.1
3	26.3	90.5
4	0.0	100.0

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	127	9	5	3	13
	1	0	0	0	0	0
	2	7	11	131	18	27
	3	0	0	0	52	7
	4	0	0	0	0	2

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	22	2	7	4	3
	1	0	0	0	0	0
	2	12	3	18	10	9
	3	0	0	7	5	1
	4	0	0	0	0	0

Testing images

Session 15

	Accuracy (%)
Training images	79.9
Testing images	39.8

Class	Recall (%)	Specificity (%)
0	94.0	93.2
1	0.0	100.0
2	96.3	83.0
3	87.8	95.0
4	16.3	100.0

Training images

Class	Recall (%)	Specificity (%)
0	41.2	87.0
1	0.0	100.0
2	65.6	42.3
3	31.6	85.7
4	0.0	100.0

Testing images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	126	10	5	1	3
1	0	0	0	0	0
2	7	9	131	8	23
3	1	1	0	65	15
4	0	0	0	0	8

Training images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	14	2	4	2	1
1	0	0	0	0	0
2	19	3	21	11	8
3	1	0	7	6	4
4	0	0	0	0	0

Testing images

Session 16

	Accuracy (%)
Training images	81.8
Testing images	45.6

Class	Recall (%)	Specificity (%)
0	97.0	89.6
1	0.0	100.0
2	94.1	94.1
3	86.5	86.5
4	32.7	32.7

Training images

Class	Recall (%)	Specificity (%)
0	79.4	65.2
1	0.0	100.0
2	43.8	63.4
3	21.1	94.0
4	15.4	98.9

Testing images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	130	15	6	3	5
1	0	0	0	0	0
2	4	5	128	7	21
3	0	0	2	64	7
4	0	0	0	0	16

Training images

Predicted Class \ Actual Class	*0*	*1*	*2*	*3*	*4*
0	27	3	14	3	3
1	0	0	0	0	0
2	7	1	14	10	8
3	0	1	4	4	0
4	0	0	0	1	2

Testing images

Session 17

	Accuracy (%)
Training images	77.7
Testing images	44.7

Class	**Recall (%)**	**Specificity (%)**
0	94.0	92.8
1	0.0	100.0
2	96.3	81.2
3	86.5	94.1
4	0.0	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	52.9	84.1
1	0.0	100.0
2	75.0	52.1
3	21.1	85.7
4	0.0	100.0

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	126	7	4	0	9
	1	0	0	0	0	0
	2	7	10	131	10	25
	3	1	3	1	64	15
	4	0	0	0	0	0

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	18	3	2	3	3
	1	0	0	0	0	0
	2	14	2	24	12	6
	3	2	0	6	4	4
	4	0	0	0	0	0

Testing images

Session 18

	Accuracy (%)
Training images	82.8
Testing images	44.7

Class	**Recall (%)**	**Specificity (%)**
0	95.5	94.3
1	0.0	100.0
2	97.8	83.4
3	75.7	97.3
4	51.0	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	58.8	85.5
1	0.0	100.0
2	75.0	43.7
3	10.5	92.9
4	0.0	98.9

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	128	8	2	3	3
	1	0	0	0	0	0
	2	5	12	133	15	14
	3	1	0	1	56	7
	4	0	0	0	0	25

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	20	1	4	3	2
	1	0	0	0	0	0
	2	13	4	24	13	10
	3	1	0	4	2	1
	4	0	0	0	1	0

Testing images

Session 19

	Accuracy (%)
Training images	82.8
Testing images	38.8

Class	**Recall (%)**	**Specificity (%)**
0	94.8	91.8
1	0.0	100.0
2	91.2	91.7
3	94.6	92.6
4	42.9	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	58.8	78.3
1	0.0	100.0
2	37.5	64.8
3	42.1	72.6
4	0.0	100.0

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	127	12	7	2	2
	1	0	0	0	0	0
	2	5	5	124	2	11
	3	2	3	5	70	15
	4	0	0	0	0	21

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	20	2	8	2	3
	1	0	0	0	0	0
	2	9	3	12	9	4
	3	5	0	12	8	6
	4	0	0	0	0	0

Testing images

Session 20

	Accuracy (%)
Training images	75.5
Testing images	36.9

Class	**Recall (%)**	**Specificity (%)**
0	88.1	91.0
1	0.0	100.0
2	94.9	76.2
3	87.8	97.1
4	0.0	100.0

Training images

Class	**Recall (%)**	**Specificity (%)**
0	44.1	76.8
1	0.0	100.0
2	62.5	43.7
3	15.8	89.3
4	0.0	100.0

Testing images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	118	12	7	1	5
	1	0	0	0	0	0
	2	16	8	129	8	34
	3	0	0	0	65	10
	4	0	0	0	0	0

Training images

		Actual Class				
		0	*1*	*2*	*3*	*4*
Predicted Class	*0*	15	2	7	4	3
	1	0	0	0	0	0
	2	19	3	20	12	6
	3	0	0	5	3	4
	4	0	0	0	0	0

Testing images

TRITA -EECS-EX-2020:342