UPPSALA
UNIVERSITET

IT 22 031
Examensarbete 30 hp
Juni 2022

Detection Of Diabetic Retinopathy Using Deep Convolutional Neural Network On Mobile Devices

Yosief Gebremariam

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Detection Of Diabetic Retinopathy Using Deep Convolutional Neural Network On Mobile Devices

Yosief Gebremariam

Diabetic Retinopathy (DR) is an ophthalmic disease that damages retinal blood vessels.DR causes impaired vision and may even lead to blindness if it is not diagnosed in the early stages. Early eye examination and detection of DR is the best solution to control the growth of this disease. DR has five stages or classes, namely normal, mild, moderate, severe, and PDR (Proliferative Diabetic Retinopathy). Normally, highly trained experts need to examine the colored fundus images to diagnose this fatal disease. This manual diagnosis (by clinicians) is tedious, error-prone, and extortionate. Therefore, various computer vision-based techniques have been proposed to detect DR and its different stages automatically. However, more research is still needed to encode the complicated underlying features. Most of the systems so far can only classify DR's different stages with very low accuracy, particularly for the early stages. Further, the diagnosis process is done only in clinics or hospitals. In this research, we are aiming to develop a mobile-based classifier model using a deep neural network, that allows the person with diabetes to have the initial analysis of DR made locally, in order to decrease the load of the eye health care professionals (opticians) when it comes to detect and classify anomalies in the retina image, and also support the professional in analysing the retina image.

Handledare: Lars Oestreicher
Ämnesgranskare: Anna Klemm
Examinator: Mats Daniels
IT 22 031
Tryckt av: Reprocentralen ITC

Abbreviations and Nomenclature

Diabetic retinopathy	DR
Proliferative Diabetic Retinopathy	PDR
Non-Proliferative Diabetic Retinopathy	NPDR
Micro-aneurysm	MA
Support Vector Machines	SVM
Probabilistic Neural Network	PNN
Convolutional Neural Network	CNN
Artificial Neural Network	ANN
Deep Learning	DL
Total cost of ownership	TCO
Mobile Inverted bottleneck convolution	MBConv
Batch Normalisation	BN
Conv	Convolution
Rectified Linear Unit	ReLU

List of Figures

1. INTRODUCTION

Diabetic retinopathy (DR) is a complication of diabetes, caused by high blood sugar levels damaging the back of the eye(retina). It is one of the major causes of blindness. DR mutilates the retinal blood vessels of a patient having diabetes. The DR has two main stages: Non-Proliferative Diabetic Retinopathy (NPDR) and Proliferative Diabetic Retinopathy (PDR) [1]. The DR in the early stages is called NPDR which is further divided into *Mild*, *Moderate*, and *Severe* stages, where the mild stage displays micro-aneurysm (MA), i.e. a small circular red dot at the end of blood vessels, and the Moderate stage, which form a flame-shaped hemorrhage in the retina when in MA get raptured. The severe stage shows the creation of new scar tissue, due to the lack of blood in the retina, which is known as intraretinal microvascular abnormalities. PDR is the advanced stage of DR which leads to neovascularization, a natural formation of new blood vessels in the form of functional microvascular networks that grow inside the surface of the retina [2].

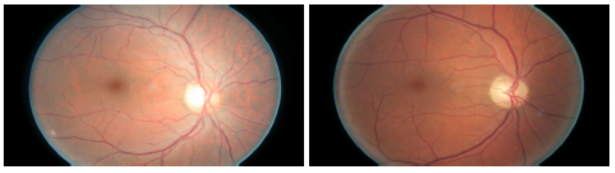
Figure 1 visually presents the different stages of DR and it is clear from the given figure that the early stages of DR look visually similar. Hence, it is difficult to differentiate and detect those early stages. Globally, the number of DR patients is expected to increase from 382 million to 592 million by 2025 [3]. A survey [3] conducted in the province of Khyber Pakhtunkhwa (KPK), and Pakistan report that 30% of diabetes patients are affected by DR and out of that 30%, 5.6% succumb to blindness. Over time, the mild NPDR develops into PDR if not controlled in the early stages. Another survey [1], conducted in Sindh, Pakistan, observed 130 patients with DR symptoms. It is reported that 23.85% of the total observed patients were DR of which 25.8% were diagnosed as PDR patients. Hence it is difficult but of utmost importance to detect the DR in the early stages to avoid the worse effect of the latter stages.

Color fundus images are used for the diagnosis of DR. The manual analysis can only be done by highly trained domain experts and is, therefore, expensive in

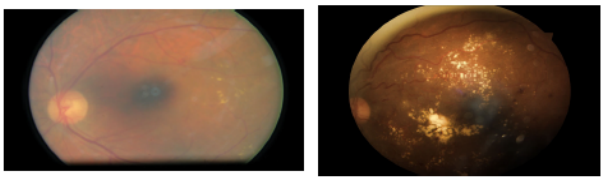
terms of time and cost. Therefore, it is important to use computer vision methods to automatically analyze the fundus images and assist the physicians/radiologists. The computer vision-based methods are divided into hands-on engineering [4][5] and end-to-end learning [6].

Many hands-on engineering and end-to-end learning-based approaches are used to detect the DR in Kaggle dataset but no approach is done to make the resulting module easily accessible with the fact that the number of diabetic patients is very high as compared to the number of eye doctors, so there is a huge necessity for developing an automated DR detection method so individuals without any clinical experience can use it to make an initial analysis of DR locally. This study focuses on implementing a mobile-based detecting model for early analysis and detection of DR.

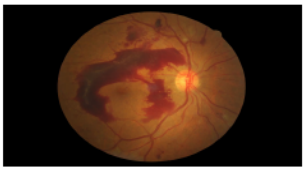
The remaining sections are organized as follows: section 2 reviews recent literature to detect and classify diabetic retinopathy. The proposed methodology is presented in section 3 while in section 4 we discuss the dataset and preprocessing methods, section 5 presents Implementation with results followed by a conclusion in Section 6.



(a) Normal (b) Mild



(c) Moderate (d) Severe



(e) PDR

Figure 1 Different stages of DR[2]

2. Related Research

In recent years, there have been many studies to automate the detection of diabetic retinopathy using AI, especially using deep learning models. As mentioned above in the introduction, studies are done either to classify or detect the stage either into binary, with DR or No DR, or into the five stages DR. The studies show that there is a good result found for classification into binary results. Prior to the rise of neural networks, most of the research on DR classification was based on traditional machine learning methods such as Support Vector Machines(SVM), Probabilistic Neural network(PNN), Bayesian Classification, random forest, and XGBoost. Since the work is done basically by extracting features of the retina, it can only be done on a small dataset. To mention some, in their paper, R. Priya et al [1], proposed to compare the performance of three ML architectures, which are SVM, PNN, and Bayesian Classification. As those architectures work with hand-on extraction of features, they only use 350 image samples and try to classify them into binary classes. Their result on accuracy is 97.6%, 94.4%, and 89.6% respectively for SVM, PNN, and Bayesian Classification. N. Kashyap et al.[7] proposed, taking the retina eye image using a phone camera with the addition of an external lens instead of using an ophthalmoscope, and once the image is taken, they carry out their feature extraction and predictions using the method of Discrete Wavelet Transform and ML Euclidean distance calculation. And their final result is Precision 63% and Recall 57% for binary classification.

The development of deep convolutional neural networks has manifested superior performance in image classification compared to previous handcrafted feature-based image classification methods. Xu et al [8] present the convolutional neural network (CNN) methodology, which belongs to the feed-forward artificial neural network (ANN), which is very similar to ordinary neural networks, to automate the classification of DR into binary classification, with DR or without DR. Since they use a relatively small dataset from the Kaggle community and CNN usually requires a large dataset in order for the model to learn enough features automatically, they use the technique called data

augmentation to increase the size of the dataset. In their paper, they conduct experiments both with and without data augmentation and the resulting accuracy is 91.5% and 94.5 respectively for testing accuracy and training accuracy. In a similar paper, Chakrabarty et al [9], proposed that they get 91.5 % for the training accuracy and 100% for the validation accuracy, still on binary classification. In their research, they explain the detailed method and stage of the CNN and they use only 30 High-Resolution Fundus(HRF) images.

But still, there is ongoing research to get a model which can detect all the five stages of DR.Rachel et al [10], focus on classifying those 5 stages of DR, and compare 3 architectures of deep neural network, Inception, VGG16, and Resnet. In their study, they use the Kaggle dataset which contains 35,000 images, although the images are unbalanced with respect to the distribution of images among those 5 classes, where most of the images is non DR.They use replacement statistics technique to balance the images in each category and after doing their experiments they conclude that VGG16 is slightly ahead from the other two CNN architectures, with the concern that it is somehow a bit hard to differentiate those on the middle. In their paper proposed by Sarki et al [12], they aim on finding an accuracy to classify those in the middle stage of the DR.Their experiment is conducted on 13 CNN architecture, which is pre-trained on a large-scale imageNet dataset, using the concept of transfer learning. After extensive experimentation, the maximum Accuracy of 86% on the No DR/Mild DR classification task was obtained for the ResNet50 model with fine-tuning. They use two datasets, the Kaggle and Messidor datasets. There is still one big problem with the use of CNN, which is the size of the model, which makes it so difficult to deploy on small devices. Patel et al [13], presented an approach that quantizes the aware training approach over the MobileNetV2 model using integer computation only instead of the original floating-point baseline models. From their experiment which is done on retinal fundus images, results with compressed size and testing accuracy for MobileNetV1 and MobileNetV2 are 94% and 96% respectively for binary classifications. Another paper focused on MobileNet is, by Patel, Sanskruti al [14], who proposed a transfer learning with fine-tuning on pre-trained CNN architectures, VGG16, and MobileNetV1 using the available retinal fundus image dataset on Kaggle to classify whether the eye has DR or not which is a binary classification. The observation from their experiment is 89.42% and 89.84 for VGG16 and MobileNetV1 respectively and the test on the accuracy by VGG16 is 89.51% and MobileNetV1 is 89.77%.

As a conclusion to the research so far made, applying a deep neural network learning method for detecting DR seems to be way more promising than the hands-on feature extraction Machine Learning method.

3. Proposed Methodology

There is a huge DL architecture that is used to solve computer vision, which mostly aims at achieving high accuracy. Where to use the model greatly affects the underlining DL architecture, that is whether the model will be used on mobile devices, served from a cloud, or real-time systems. Below we will see the difficulties with standard CNN and our chosen architectures.

3.1 Basic CNN structure

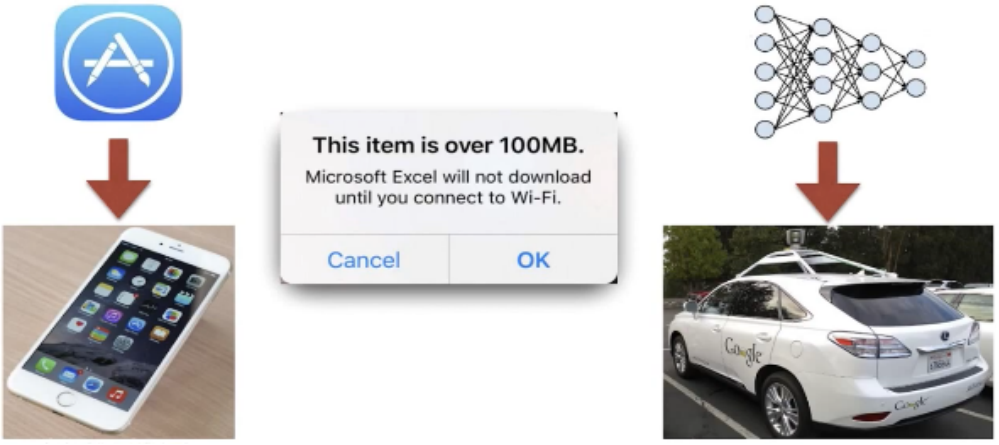
A Convolutional neural network (CNN) is a DL architecture, mostly used within the field of computer vision and image analysis, which is made up of stacking several convolution layers, where each layer contains a number of features extracting image kernels for object detection and classification. On the top of each convolution layer, there is also a relu function, with the primary role being to change the linear output feature image from the Conv layer into a non-linear feature image. Furthermore, there is also commonly a pooling layer, which reduces the size of the feature images and forwards them to the next Conv layer or to the final fully connected layer. The idea of the pooling layer is that, once the feature is extracted, the result contains both important features together with some additional unnecessary information. Thus, pooling is used to extract or reduce the image to the important extracted features only. So the general trend has been to make deeper and more complicated networks in order to achieve higher accuracy [19].

3.2 What are the challenges related to most CNN networks?

As computer vision is becoming a common solution in many areas of our lives, the demand for an effective and compact model becomes higher and higher. The traditional deep neural networks have several disadvantages while developing and more even during deployment. To mention some of the *challenges*:

1. Model Size.

The larger the size of the model, the more computer space it needs to deploy, and furthermore the over-the-air updating is also going to be time-consuming.



The app icon is in the public domain
Phone image is Licensed under CC-BY 2.0

The image is Licensed under CC-BY 2.0

Figure 3.1. Related to small devices model problem

Small devices have an issue related to space and battery, where the neural network model usually requires more energy and space.

2. Speed.

A model with high accuracy will also become a large model. Larger models basically require longer training time, and likewise make the experiment more difficult, for example, ResNet.

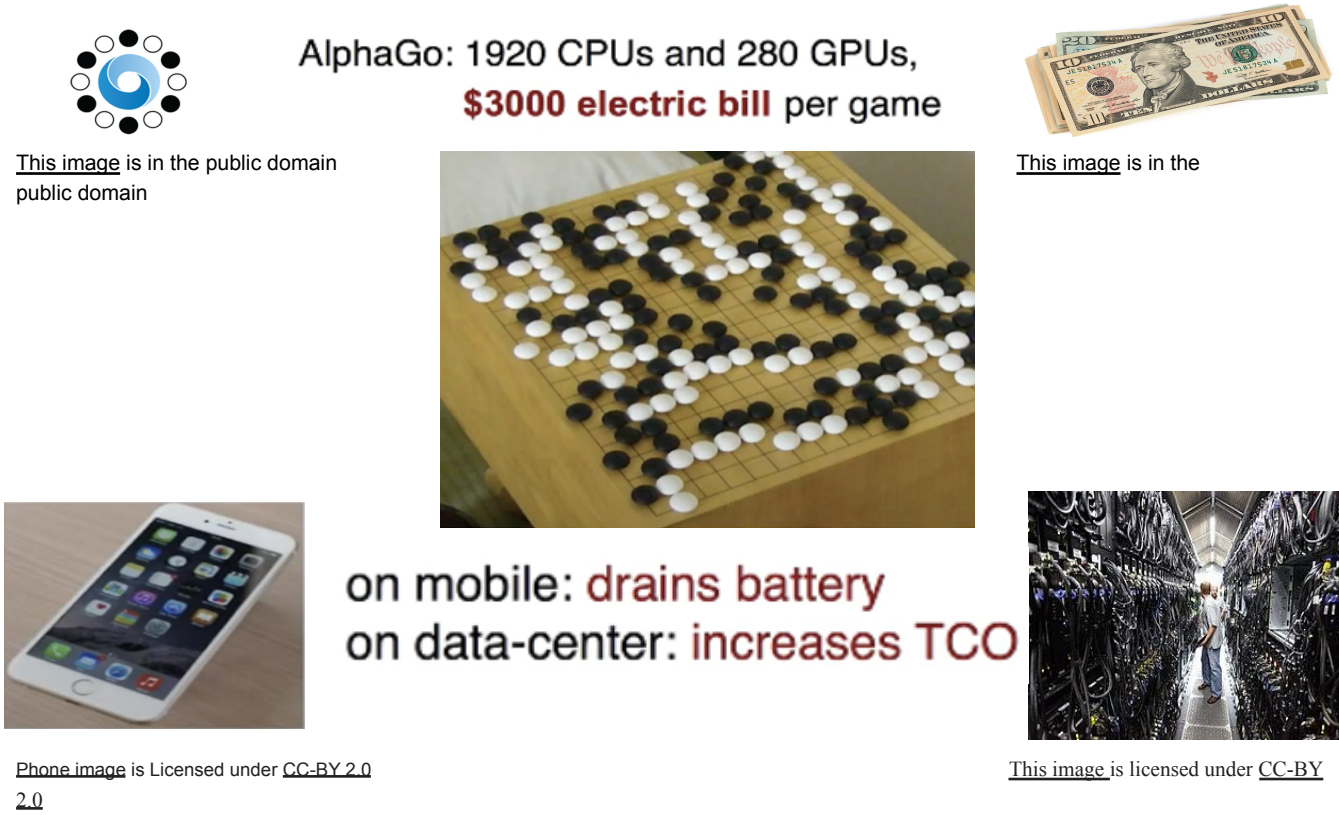
	Error Rate	Training time
ResNet18	10.76%	2.5 days
ResNet50	7.02%	5 days
ResNet101	6.21%	1 week
ResNet152	6.16%	1.5 weeks

Table 1. Long training time limits ML researcher's Productivity [16]

From table 1 we can see that the more time spent training the model the accuracy of the model will increase, but it is at the expense of time.

3. Efficiency

General deep neural network is not efficient in terms of its energy usage, which basically adds to the increasing greenhouse effect[16].



This image is in the public domain public domain

This image is in the

Phone image is Licensed under CC-BY 2.0 2.0

This image is licensed under CC-BY

Figure 3.2: Side effect of the large model in relation to energy usage.

3.3 Alternatives to the large size deep neural network?

Some necessary points to be considered while using a Deep Neural network are

1. High performance (Accuracy).
2. Efficiency (maximum productivity with less minimum effort).
3. Scaling -good scaling techniques by adding more resources to the model, which can be:

- width scaling, increasing the number of filters in each layer
- Depth scaling, adding more layers
- Resolution scaling, or increasing input resolution

3.3.1 Overview of the chosen Model architectures

To select the best architecture for a specific deep learning problem is a crucial design choice since the architecture used affects the classification performance level that may be attained.

However, for the medical imaging domain in particular, the declared performance of these architectures on large-scale natural image classification may not always be the most relevant, due to other considerations. For one, the relatively small quantities of medical image data available in some areas may lead to overtraining and/or difficulties with convergence during training with more-sophisticated and higher-capacity models. As such, other than the careful application of transfer learning (covered later), older and simpler architectures may sometimes be favored for particular applications. For example, the VGGNet architecture remains exceptionally suited for the extraction of intermediate features [10], while requiring relatively more weight parameters than other popular architectures [11].

Another important consideration for the model architectures would be the number of computing resources required, which is relevant for deployment on consumer devices such as smartphones, embedded systems, and possibly less powerful hardware in under-resourced situations. In general, the fewer the number of weight parameters involved in the model architecture, the quicker the inference. If the inference time is sufficiently quick, the real-time analysis further becomes possible. Lightweight model architectures such as MobileNet [22] and EfficientNetB0 [23] have been designed for devices with limited computing power, although a compressed standard model can also be achieved through pruning and parameter quantization [25]. From their paper Howard et al [22], the architecture of the MobileNet targets primarily very small, low latency models that can be easily matched to the design requirements for mobile and embedded vision applications. In their paper, Tae et al [23], suggest that it is critical to balance all the dimensions of the network width/depth/resolution with a compound scaling method, which works well on existing MobileNets.

3.3.2 **MobileNets:** Depthwise Separable Convolution

Unlike standard convolution, the model of MobileNet is based on *depthwise separable convolution* which is a form of refactoring of the standard convolution into a combination of a depthwise and pointwise convolution. This refactoring, of the standard convolution into a depthwise Convolution of filters in one layer and combining them in pointwise Convolution on the next layer, has drastically reduced both computation speed and model size.

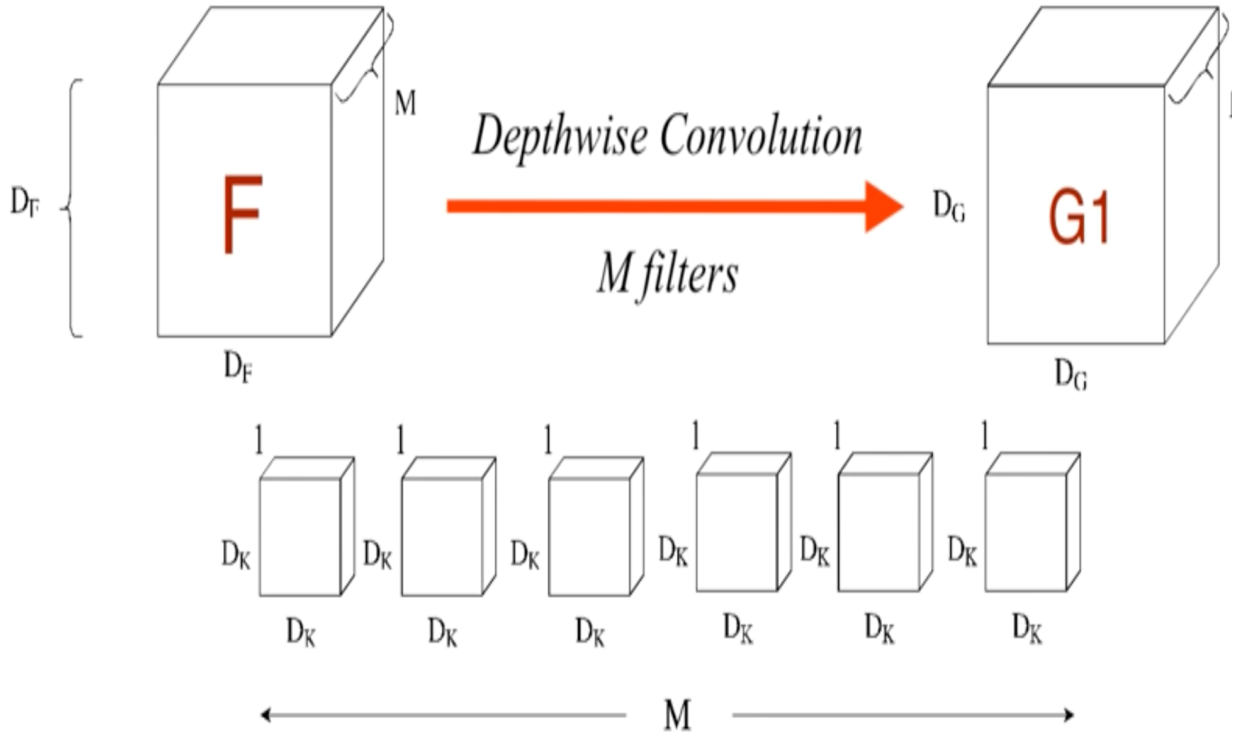


Figure 3.3. Depthwise Convolution: where each channel is scanned separately[17]

MobileNet spends 95% of its computation time in 1 × 1 convolutions as in Figures 4 and 5.

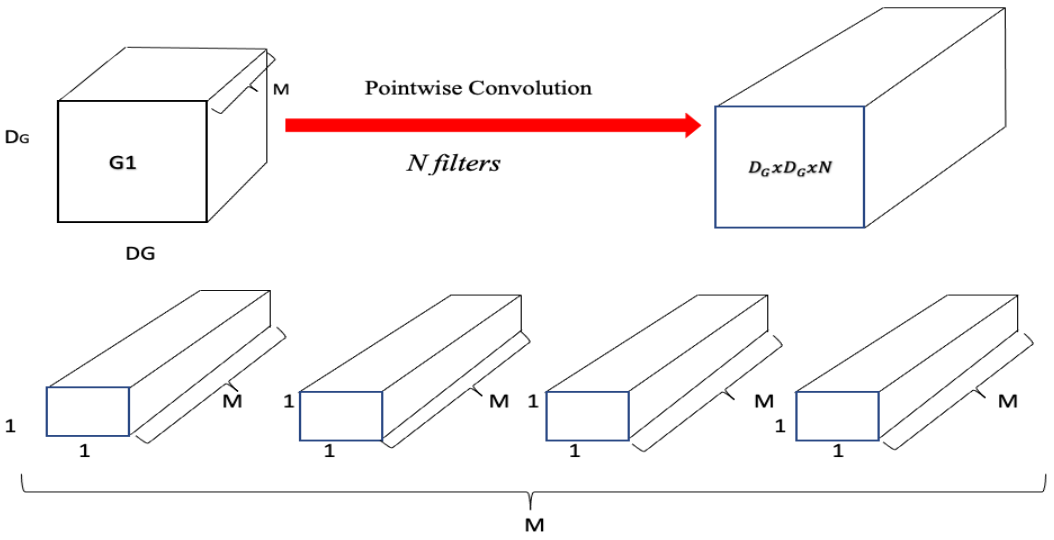


Figure 3.4: Pointwise Convolution: combination stage[17]

The cost of the computational depthwise separable convolution is the sum of computation for depthwise convolution and pointwise convolution, that is:

$$D_K . D_K . M . D_F . D_F + M . N . D_F . D_F \quad \text{.....(1)}$$

If we take the computational cost of depthwise separable over the standard convolution, eq 3 shows the difference in the computation that the depthwise separable have:

$$= \frac{D_K . D_K . M . D_F . D_F + M . N . D_F . D_F}{D_K . D_K . M . N . D_F . D_F} \quad \text{.....(2)}$$

$$= \frac{1}{N} + \frac{1}{D_K^2} \quad \text{.....(3)}$$

The MobileNet uses 3 × 3 depthwise separable convolutions, and this requires between 8 to 9 times less computation power than standard convolutions and with only a small reduction in accuracy.

Moreover, the mobileNet introduces a set of two hyper-parameters i.e width multiplier **α** and resolution multiplier **ρ**) in order to build a specific use case or

application that may allow the model to be very small in size while still having low latency, **i.e** the time one has to wait to get the result during the application of the model. This is also directly tied to the real-time performance of the system, a model that can be easily matched to the design requirements for mobile and embedded vision applications. So the network focuses on achieving a decrease in latency (less latency is better) and increasing accuracy by trying to have a small model. Using the two model shrinking hyper-parameters that are available i.e width multiplier and resolution multiplier, the model can still reduce the network's computational power. We will start by looking at some of those hyper-parameters:

1. Width Multiplier: its role is to uniformly thin the network at each layer. It is applied to each input channel M with becoming αM and the number of output between channels N becomes αN,, where $\alpha \in (0, 1]$ with typical settings of 1, 0.75, 0.5, and 0.25. $\alpha = 1$ is the baseline MobileNet and $\alpha < 1$ is the reduced MobileNets. The width multiplier has the effect of reducing computational cost and the number of parameters quadratically by roughly α^2

$$DK.DK.\alpha M.DF.DF + \alpha M.\alpha N.DF.DF \quad \text{.....(4)}$$

2. Resolution Multiplier: This is applied to the input image and the subsequent internal representation of every layer is reduced by the same multiplier. $p = 1$ is the baseline MobileNet and $\rho \prec 1$ are the reduced computation of MobileNets. The resolution multiplier has the effect of reducing computational cost by $\boldsymbol{\rho^2}$**.**

$$Dk.Dk.\alpha M.\rho DF.pDF + \alpha M.\alpha N.\rho DF.\rho DF \quad \text{..... (5)}$$

The 3x3 Conv in standard Conv is subdivided into 3x3 depthwise Conv followed by BN and Relu, and followed by 1x1 Conv followed by BN and ReLU. The below table shows the body architecture of MobileNet.

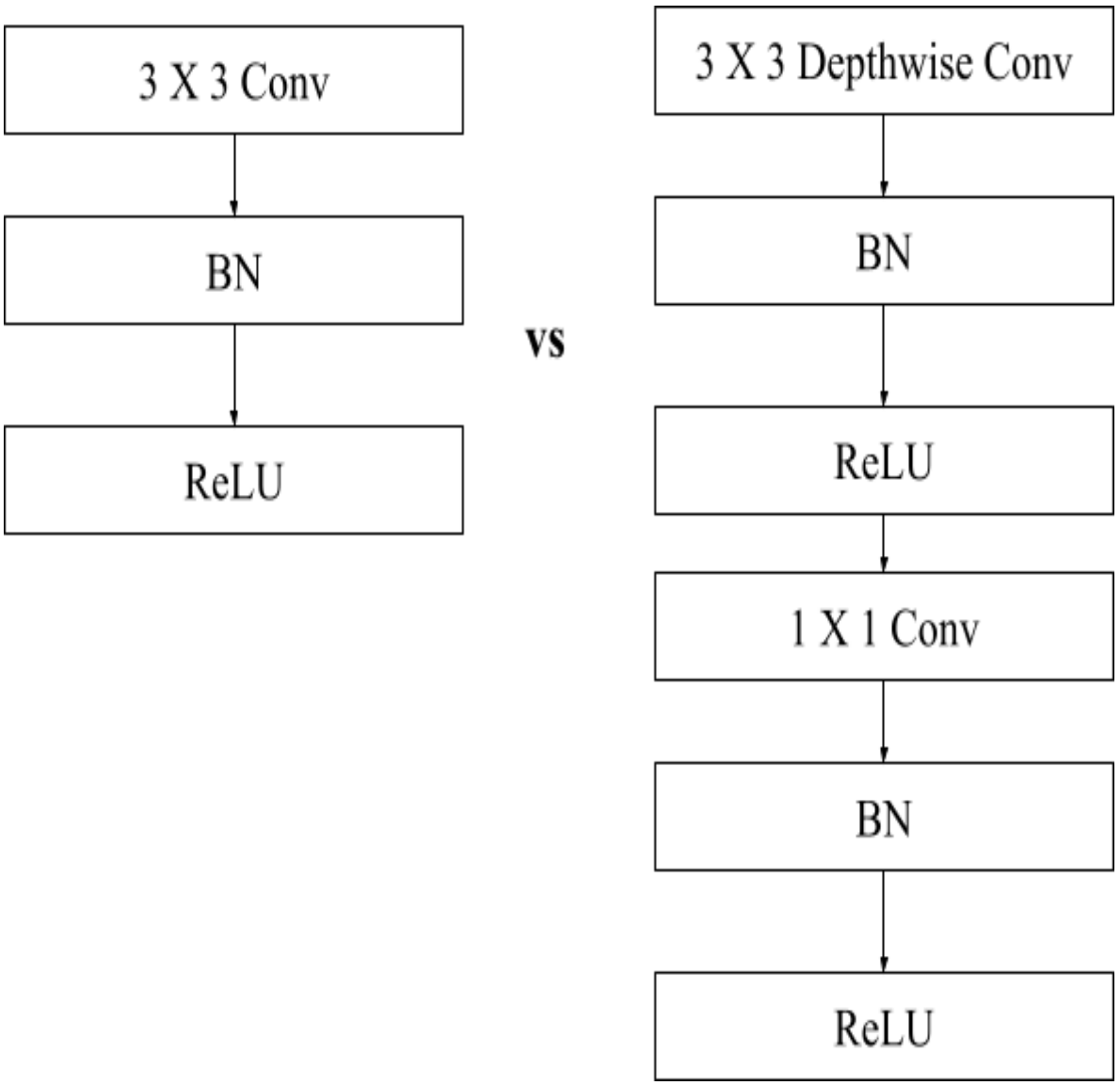


Figure 3.5: Standard Conv vs depthwise Conv [22]

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Table 2. MobileNet body architecture [22]

Tables 3,4, show a comparison of computation and parameters number of MobileNetV2 with other types of ordinary CNN and Conv MobileNet too.

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0MobileNet-244	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG16	71.5%	15300	138

Table 3 . MobileNet Comparison to Popular Models[22]

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
MobileNet	70.6%	569	4.2
Conv MobileNet	71.7%	4866	29.3

Table 4. Depthwise Sparable vs Full Convolution MobileNet[22]

3.3.3 EfficientNetB0

The efficientNet takes the idea of scaling further in order to enhance the training, with a method introduced in the mobilenet to achieve a better scaling method. Here the idea of the *compound scaling method* has been introduced to create a better architecture. Intuitively, the compound scaling method makes good sense because if the input image has a higher resolution, then the network needs more layers, that is, it needs more depth, i.e., more layers in order to

capture more complex features. Hence more channels will also be needed to capture detailed fine-grained patterns on the bigger image.
The compound scaling method uniformly scales all dimensions, depth, width, and resolution, using a simple yet highly effective compound coefficient. In order to pursue better accuracy and efficiency, it is critical to balance all dimensions of network width, depth, and resolution during ConvNet scaling, and surprisingly such balance can be achieved by simply *scaling each* of them with a constant ratio.

Based on their observation, Tau et al. [18], propose a simple yet effective *compound scaling method***,** meaning uniformly scales the depth, width, and resolution with a set of fixed scaling coefficients.

$$\text{depth: } d = \alpha^{\phi}$$
$$\text{width: } w = \beta^{\phi}$$
$$\text{resolution: } r = \gamma^{\phi}$$
$$\text{ds.t.} \quad \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \ \beta \geq 1, \ \gamma \geq 1$$

For example, if we want to use *2 ^N* times more computational resources, then we can simply increase the network depth by *α^N,* width by *β ^N*, and image size by *γ^ N*, where *α, β, γ* are constant coefficients determined by a small grid search on the original small model. It is based on a Neural Architecture search to design a new baseline network and use the scaling coefficients to get the different family models. While increasing those three parts of a network we need to make sure that we have always checked that the *Memory and FLOPs (computational)* are below the given target memory and target number of flops. *Floating-point operations per second* (FLOPS, flops, or flop/s) is a standard measure of computer performance, useful in fields of scientific computations that require floating-point calculations.

Figure 3.6 shows the building block taken from the MobileNet, which is MBConv, which is a composite of three components. Those are 1x1 Conv, Depthwise Separable, and Result connection.

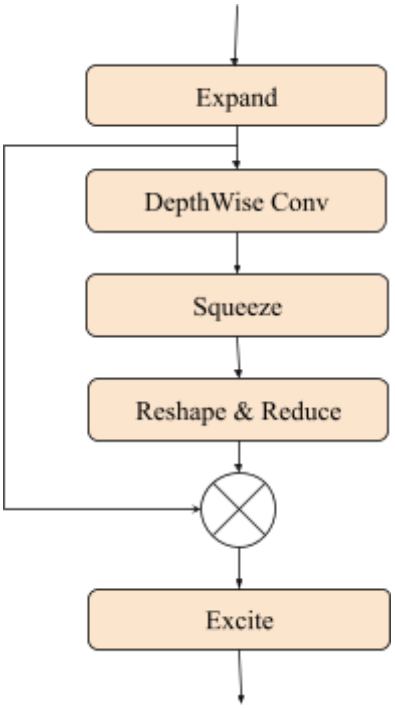


Figure 3.6: MBConv, Basic Building block of EfficientNet [26]

The overall architecture can be divided into seven blocks which are shown in different colors in the figure below. The basic building block of the network is MBConv (mobile inverted bottleneck convolution). Each MBConvX block is shown with the corresponding filter size and the X=1 and X=6 denote the standard ReLU and ReLU6 activation function respectively[26].

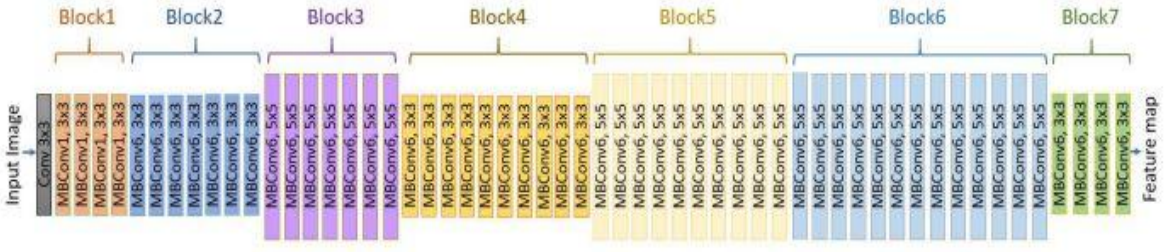


Figure 3.7: Architecture of EfficientNetB7 with MBConv as basic building blocks[26]

4. Dataset setup and methods used

To have a good model which can classify a given set of domains, it greatly depends on the amount and type of dataset and the preprocessing method used to improve the quality and cleanliness of the data. Here we discusse our datasets and preprocessing methods we use.

4.1 DataSet Description

We have used two separate datasets that have different sizes and types of images. Both datasets are publicly available on the Kaggle website [20,21]. The dataset is unbalanced with most of the images belonging to the first and third classes as shown in the diagram below since most of the images are from Non-diabetic retinopathy stages. The first dataset contains 3662 sample images, where the images are preprocessed by applying Gaussian filters and resizing them to 224 x 224.

	Stages of DR	Number of images
1	Non-Diabetic Retinopathy	1805
2	Mild	370
3	Moderate	999
4	Severe	295
5	Proliferative	193

Table 5: Number of images in the dataset from Kaggle website [20]

The distribution of the images can be seen in the graph below also.

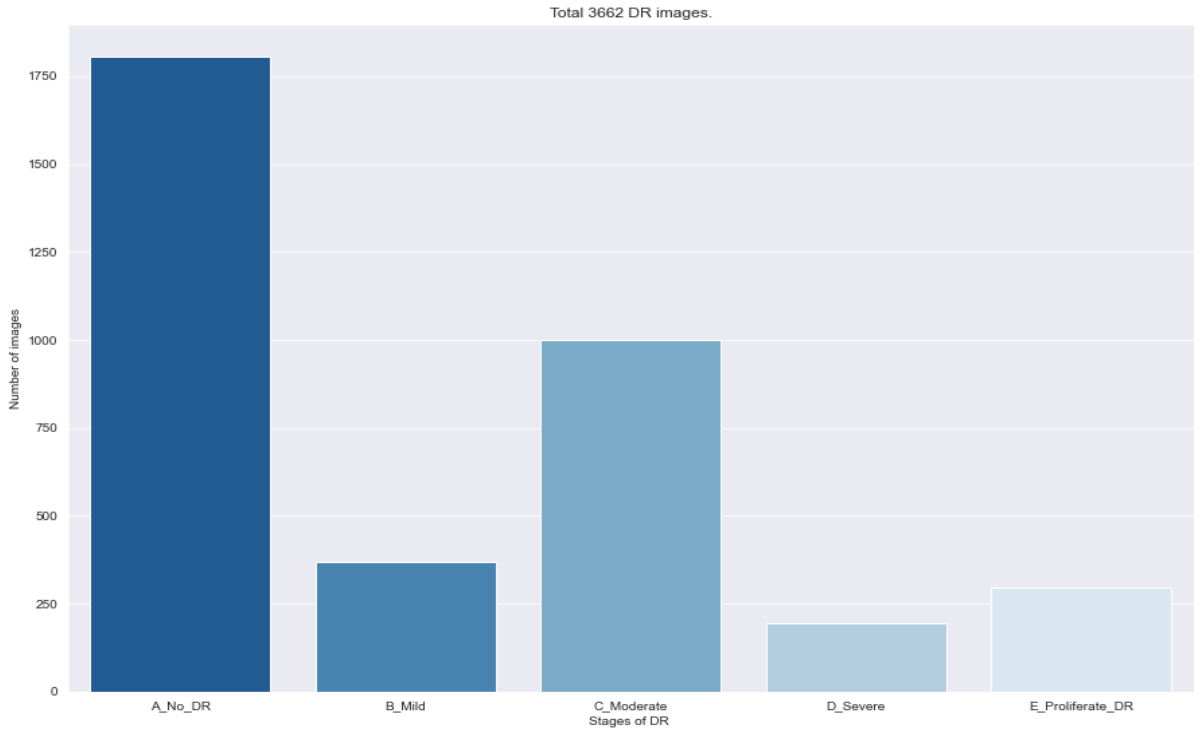


Figure 4.1: Total 3662 DR Unbalanced dataset

The second dataset contains a very large number of samples, which is around 35,126 images, where each of the images is unprocessed and marked with the left or right eye and its level of disease: 0, 1, 2, 3, 4. The distribution of each sample stage is shown in the diagram below [figure 4.2].

	Stages of DR	Number of images
1	Non-Diabetic Retinopathy	25810
2	Mild	5292
3	Moderate	2443
4	Severe	873
5	Proliferative	708

Table 6: Number of images in the dataset from Kaggle website [21]

The distribution of the image can be seen in the graph below also.

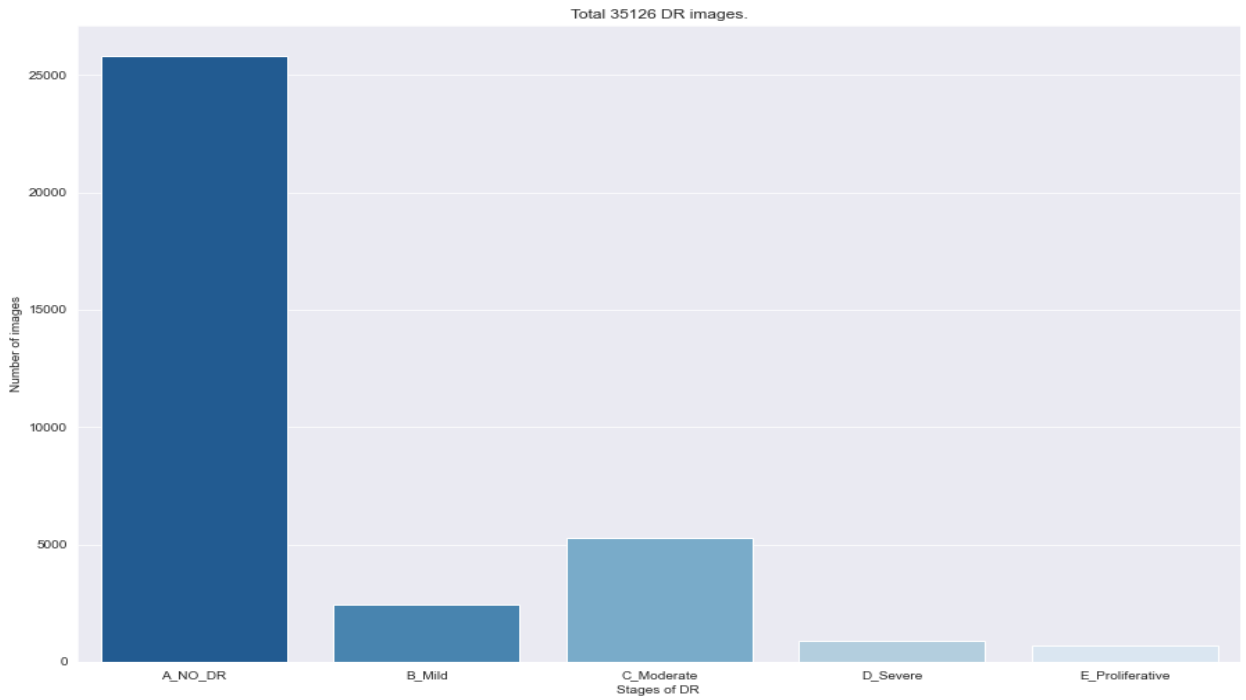


Figure 4.2: Total 35126 DR Unbalanced dataset

Skewed datasets may cause the network to over-fit to the most prominent class in the dataset. Large datasets are often massively skewed and need some kind of balancing preprocessing. Data augmentation is the first suggestion that is mostly applied to this kind of dataset. We perform our training, testing, and evaluation with 1000 samples from the first relatively small dataset and 2000 samples from the second, larger dataset. To balance each sample within each stage, we use data augmentation, (i.e), using image transformation techniques, with the transformation parameters such as flipping (horizontal), rotation, shifts (width and height), and zooming.

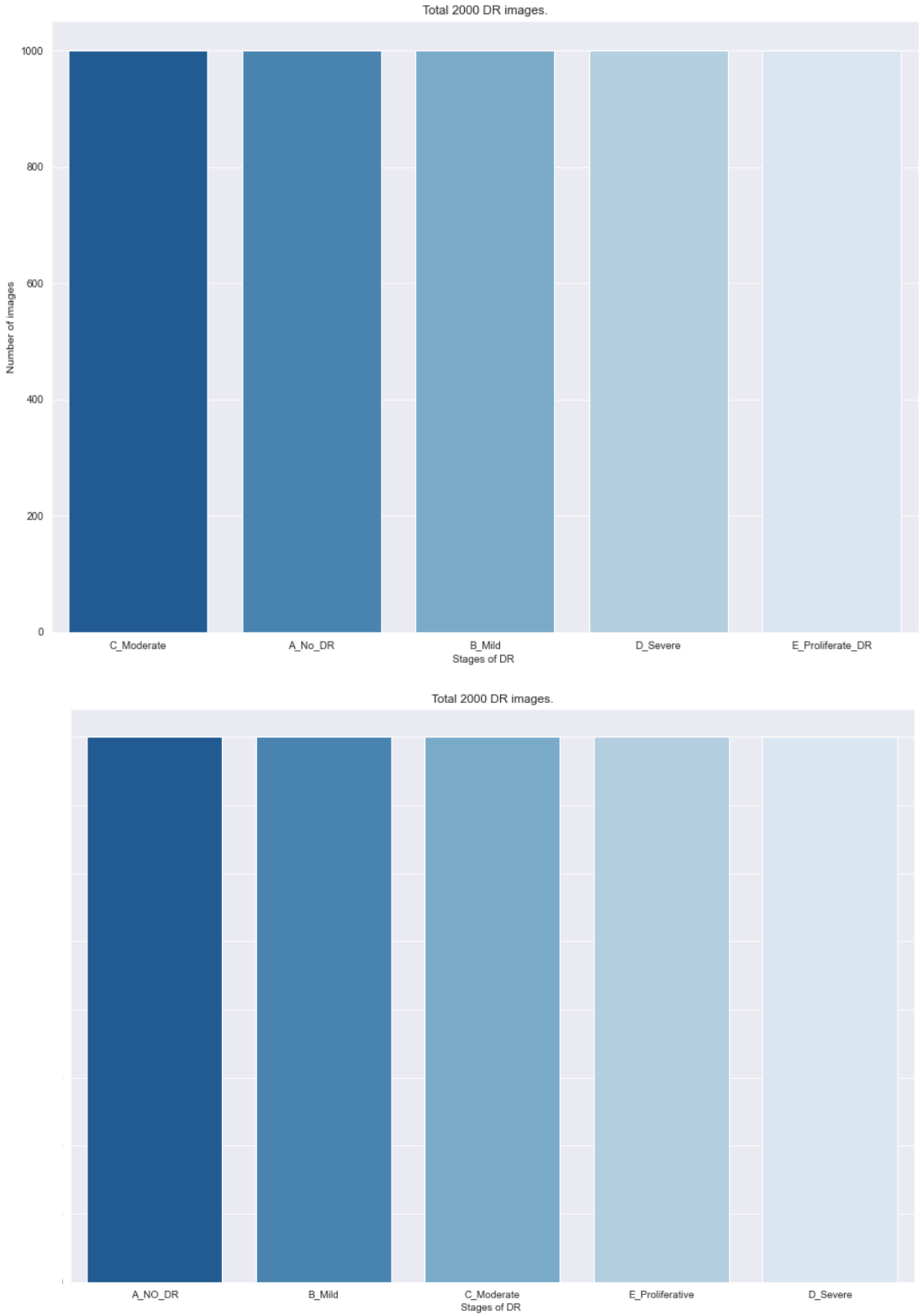


Figure 4.3: Balanced dataset using data augmentation

Finally, we merge the two datasets, those with less number samples, and make a dataset without data augmentation, finally creating a dataset with 1000 for each stage.

4.2 Image Preprocessing

The biggest problem with the detection of DR in different stages is that there are overlapping symptoms, where the symptom shown in mild, will also be visible in the moderate stage, in addition to the moderate specific symptoms and this makes the classification a bit hard especially on the last stages. As shown in figure 4.4, the symptoms that the model used as a feature to classify the images are:

1. Micro-aneurysm or Hemorrhage
2. Hard Exudates
3. Intra-Retinal Microvascular Abnormality (IRMA)
4. Neovascularization
5. Preretinal Hemorrhage

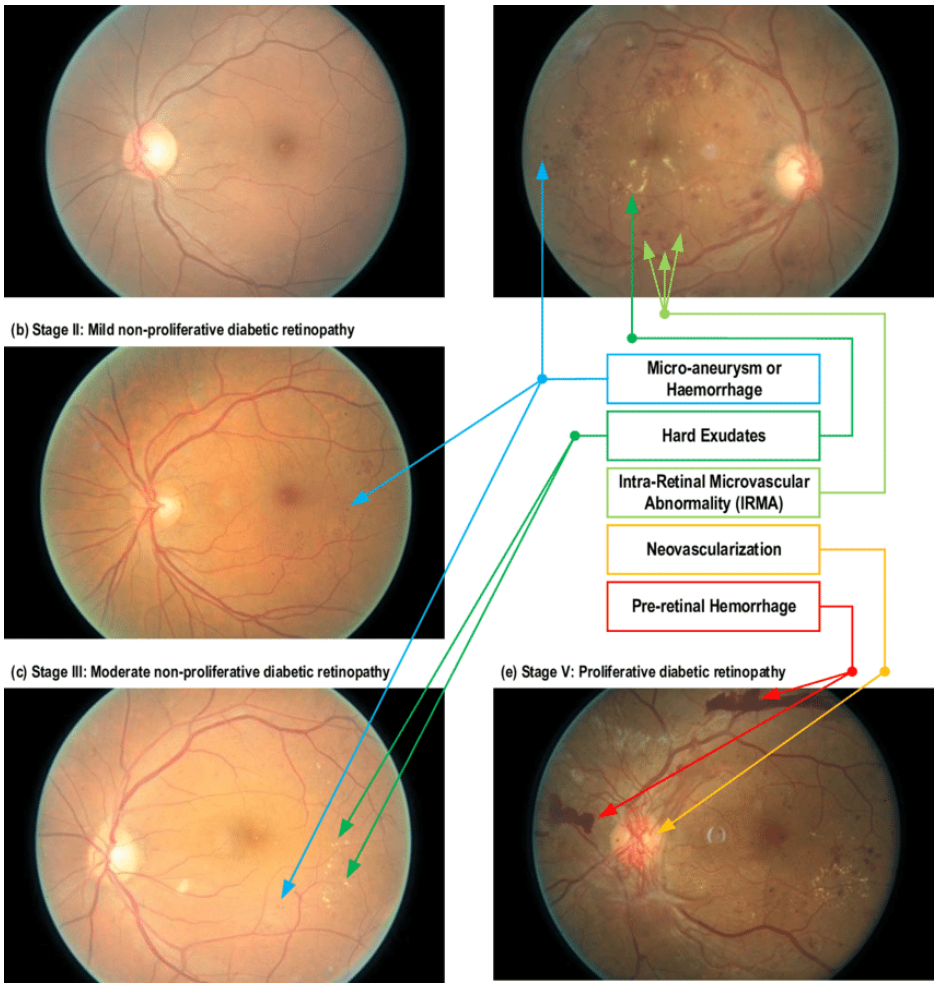
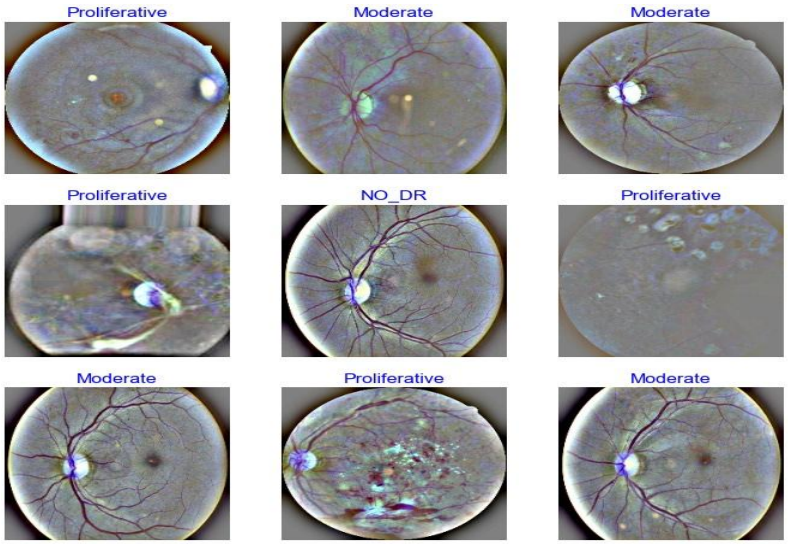


Figure 4.4: Specific symptoms of the 5 stages of DR [19,27]

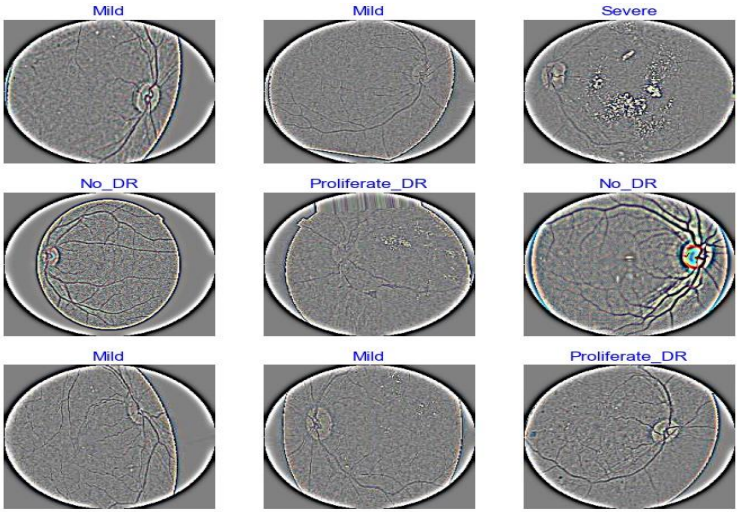
As an additional problem, the images come with many different lighting conditions, some images are very dark and difficult to visualize, and also different sizes, which add more difficulties during processing.

So we try to make different preprocessing techniques for those images before we feed them to the model to train it.

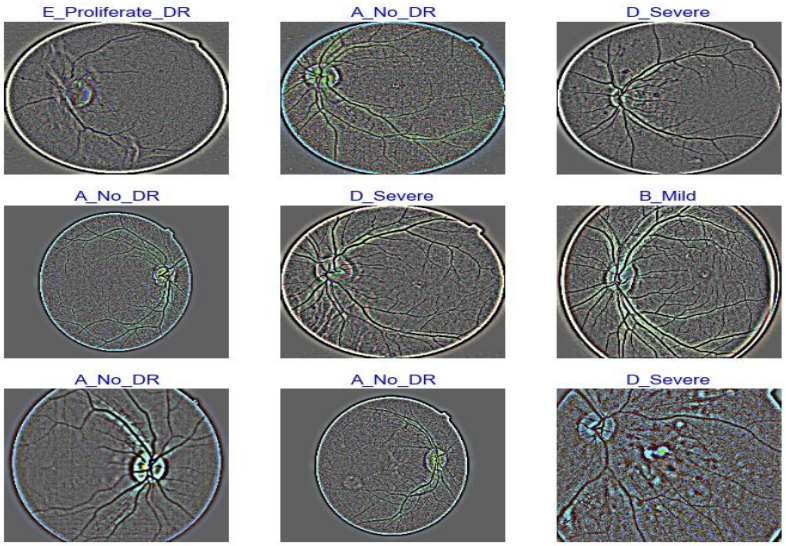
1. We resize all the images into 224 x 224, which will result in all the images having the same size, especially for the second large dataset [21], where the images have different sizes.
2. To improve the lighting condition which greatly affects and improves to see details of the eye, we use the idea used by Ben Graham (last competition's winner, in the Kaggle competition)[28], which is basically converting the images into grayscale images first and then use Gaussian method with a different sigmax value, which affects the lighting. Below are some of the results after we apply image preprocessing with their applied function description.



Basic Gray scale transformation



Gray and Circled Transformation



Twice Grayscale transformation
With Gaussian filter.

Figure 4.5:Sample of different transformations.

4.3 Transfer learning

Building and training a deep neural network from scratch takes a lot of time. Transfer learning is a technique to transfer a model parameter pre-trained in a common public dataset (e.g. ImageNet) to the new image domains. The most prominent application of transfer learning has perhaps been in the finetuning of models that have already been pre-trained on another classification task. The reasoning behind such transfer learning in our study is that the retinal image domain and the natural image domain share some similarities, especially for the universal lower-level features such as corners and edges. Therefore, the parameter weights from a natural image classification task should then serve as a good initialization for retinal image classification.

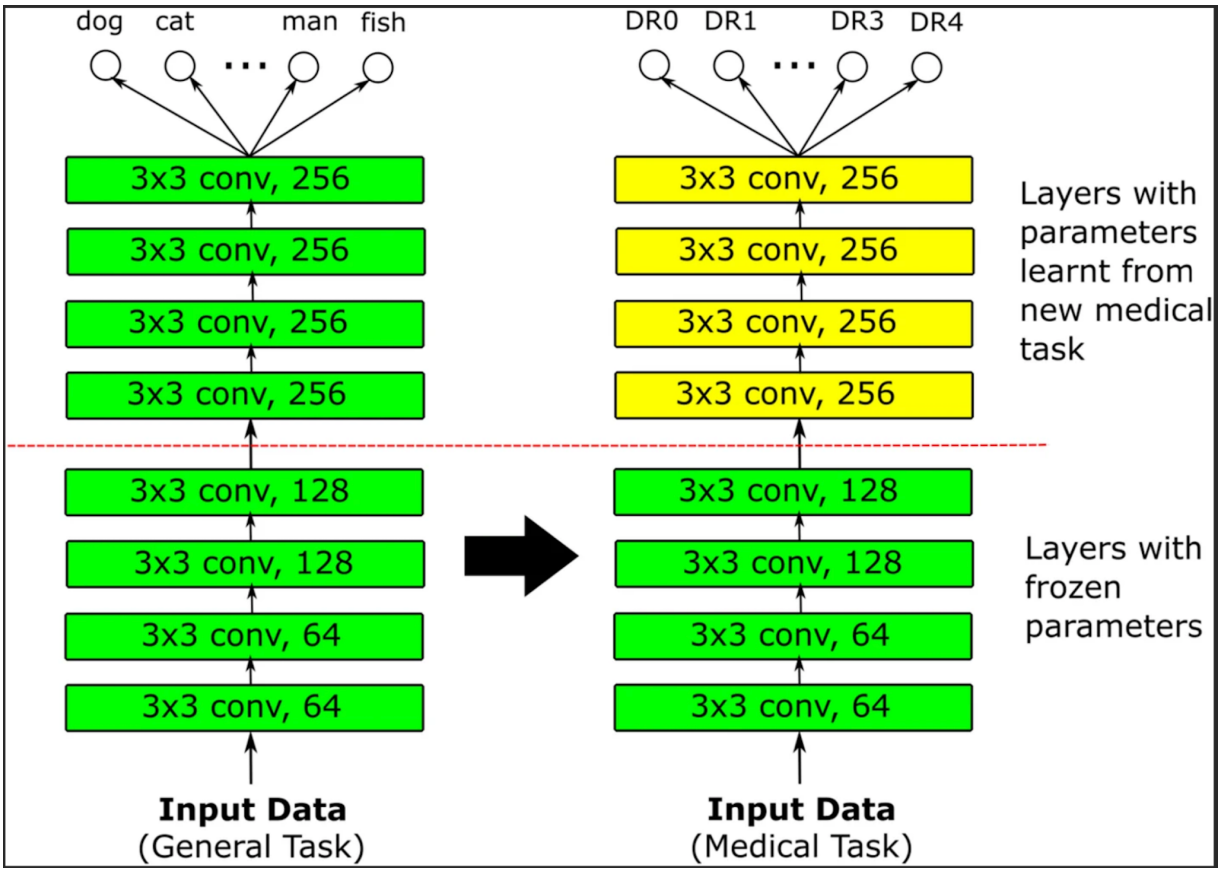


Figure 4.6: Transfer learning by freezing the top layers[15]

A previous survey on transfer learning in the medical domain by Tajbakhsh et al. [25] suggests that although the use of pre-trained weights made DL models

more robust to the size of training sets, the optimal selection of layers to fix depends on the task at hand and has to be empirically determined.

In both our models we use the *freezing of the convolutional base* and then add a single *batch normalization layer*, a fine-tuning *dropout layer*, and two fully connected layers with *activation functions* ReLU and softmax. Then retraining the final classification layer.

The advantage of using transfer learning is that it does not require a large processing speed and time to train the model. The frozen layers are already trained intensively to classify several images, which are mostly found in imageNet so the required training is for those which are added at the bottom of those already trained layers.

Layer (type)	**OutputShape**	**Parameter Number**
Input_1(InputLayer)	(None,224,224,3)	0
MobileNetV2/EfficientNetB0(Global_max_pooling2d)	(None,1280)	0
batch_normalization_1	(BatchNorma(None,1280)	5129
fcDencse(Dense)	(None,256)	327936
DropOut	(None,256)	0
fcOutput(Dense)	(None,5)	1285

Table 7: Layers add to the freezing layer for both the architectures

	Total parameters	Non-Trainable parameters	Trainable parameters
MobileNetV2	2,592,325	2,260,544	331,781
EfficientNetB0	4,383,912	4,052,131	331,781

Table 8: Parameter summary for both MobileNetV2 and EfficientNetB0

5. Implementation, Results and First Discussions

5.1. Implementation Details

The data pre-processing (Image Processing) is performed using Python's CV2 library's miscellaneous routines. The model is trained on a machine with Intel(R) Core i7 6-core processor 2,6GHz CPU,32GB RAM. CPU is used as the interface with Python's Keras Library (TensorFlow in the backend).

The dataset is split into 80%,10%, and 10% of training, testing, and validation data respectively. Each model is trained by running up to 40 epochs with a controlling function that if the model does not show any improvement, it will automatically halt and get terminated.

5.2 Intermediate model training visualization

Taking a look at the structure of the model, in particular, the EfficientNetB0 building layer, with the addition of our 4 layers, has a total of 242 layers where those layers can be grouped into 7 stages of similar structure. Each stage is composed of *Dwconv (depth width Conv), bn (batch normalization), activation, squeeze, reshape, reduce expand, excite, conv and bn layers*. At each stage, during the training, the model tries to learn specific features of the eye .

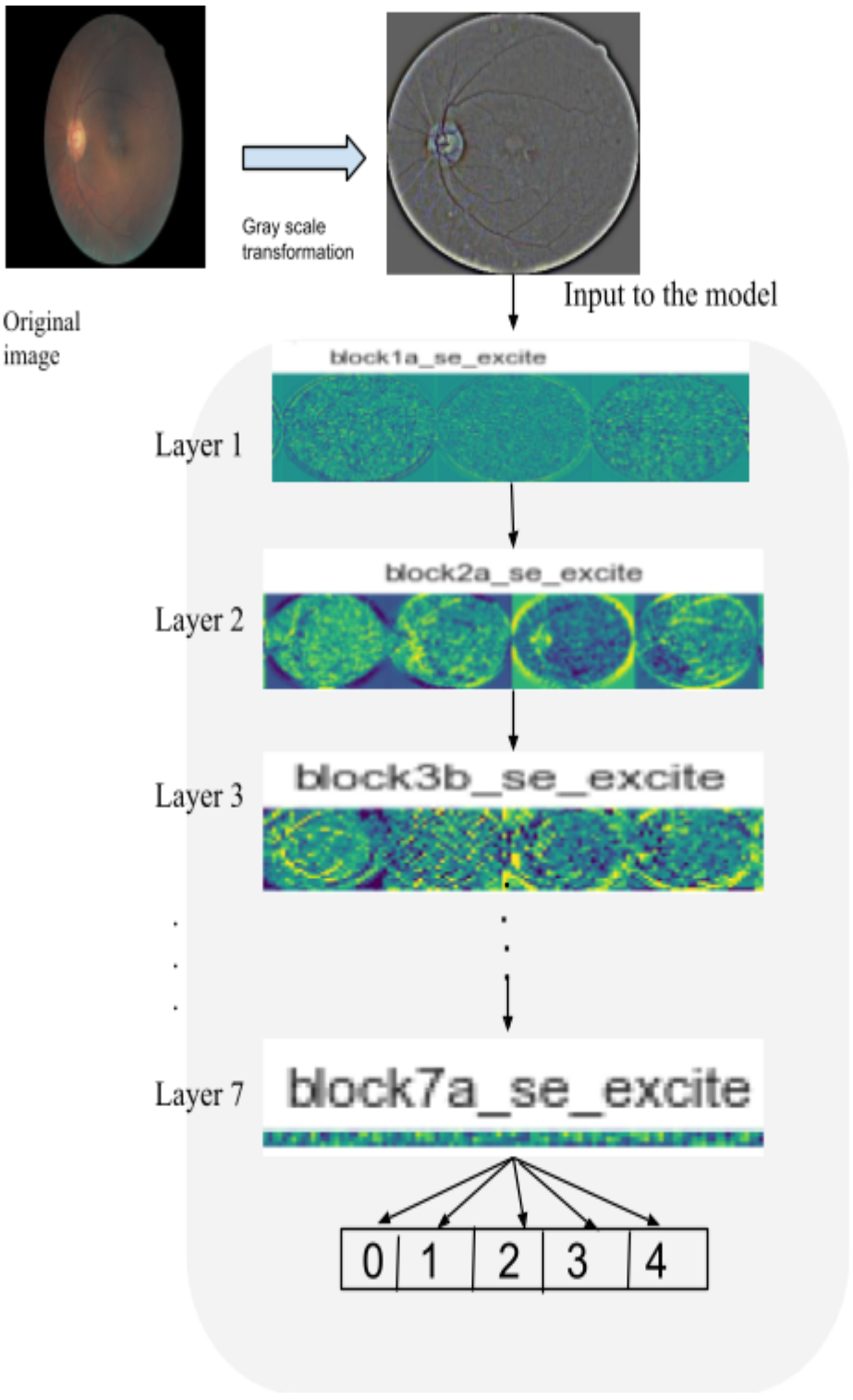


Figure 5.1: EfficientNet Internal feature extraction

5.3 Results

Our experiment is initially conducted with images from both datasets and with the help of data augmentation to avoid overfitting during the training time. Next, we try to merge the stages of DR into 3 stages and train the model with the above dataset. Furthermore, we do experiments with the preprocessed images using the Gaussian filter function with different parameters. Finally, we merge the two datasets in order to avoid data augmentation and preprocess those images. Below are the four experimental results and discuss their performance and accuracy.

Performance parameters.

To quantitatively evaluate the proposed model we use accuracy, precision, F1-score, and confusion matrix performance.

Accuracy: a ratio of correctly predicted observations to total observations.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Where TP - True positive
TN - True Negative
FP - False Positive
FN - False Negative

Precision: the ratio of correctly predicted positive observations to the total predicted positive observations.

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

Recall (Sensitivity): the ratio of correctly predicted positive observations to all observations in the actual class.

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

F1-score:- is the weighted average of Precision and Recall.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

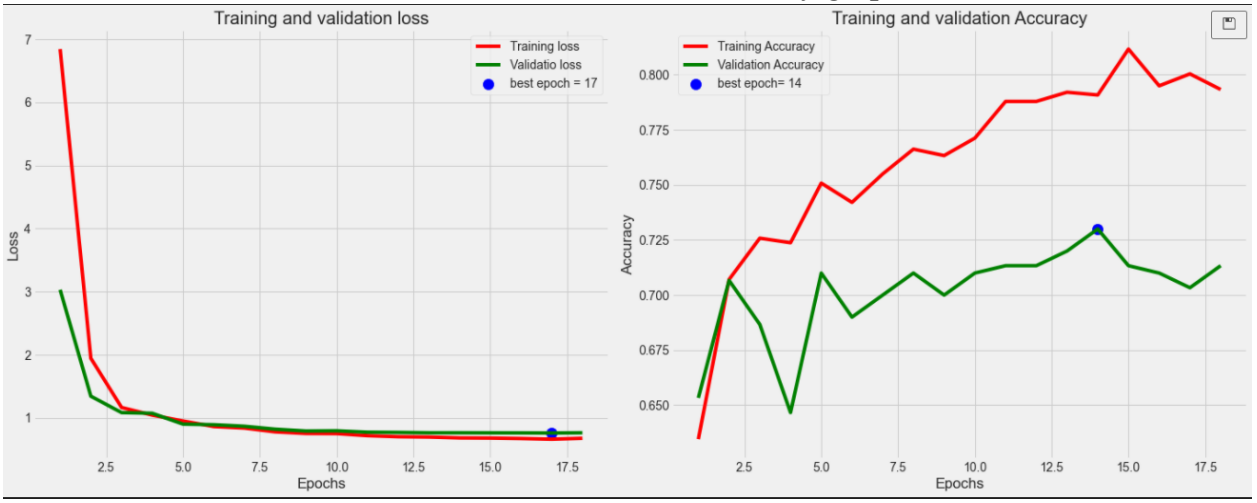
Confusion matrix: a summary of prediction results on a classification problem.

5.3.1 Model trained with an augmented dataset to classify 5 stages.

In this training phase, we make the first data augmentation to make the number of samples in each stage of DR to be 1000 images and run the experiment for both

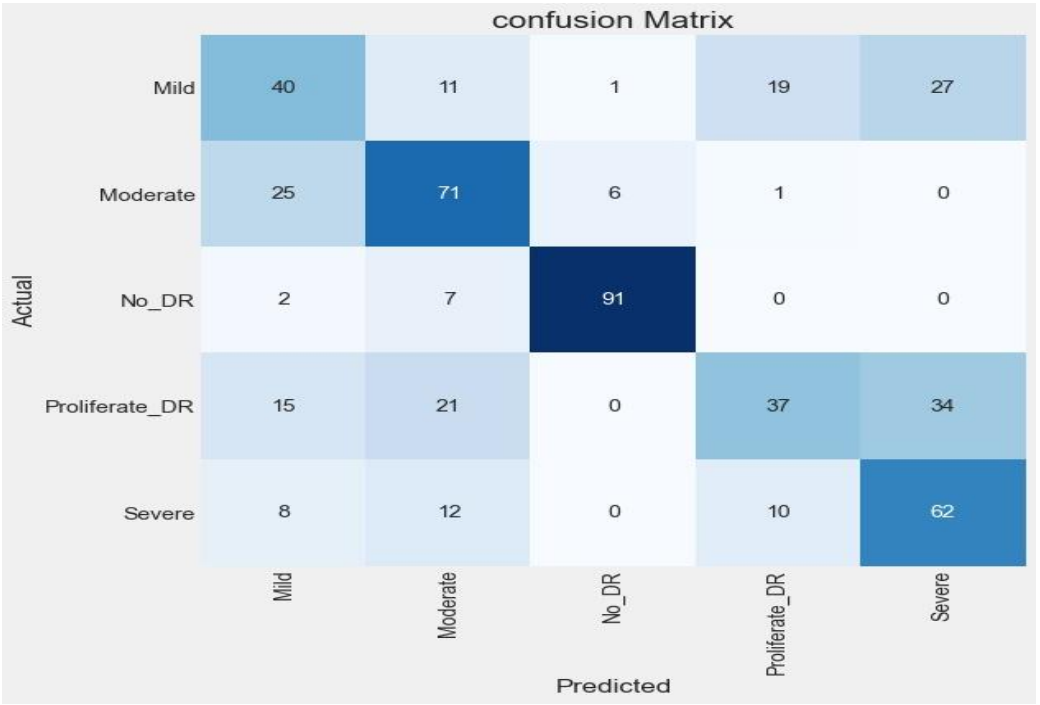
MobileNetV2 and EfficientNetB0. As mentioned above the EfficientNetB0 is built by taking the advantage of those building blocks of MobileNetV2, so it has more accuracy than MobileNetV2 with all our experiments. Below is the figure showing the improvement of models during testing and validation and the confusion matrix for both of the models.

MobileNetV2 loss and accuracy graph

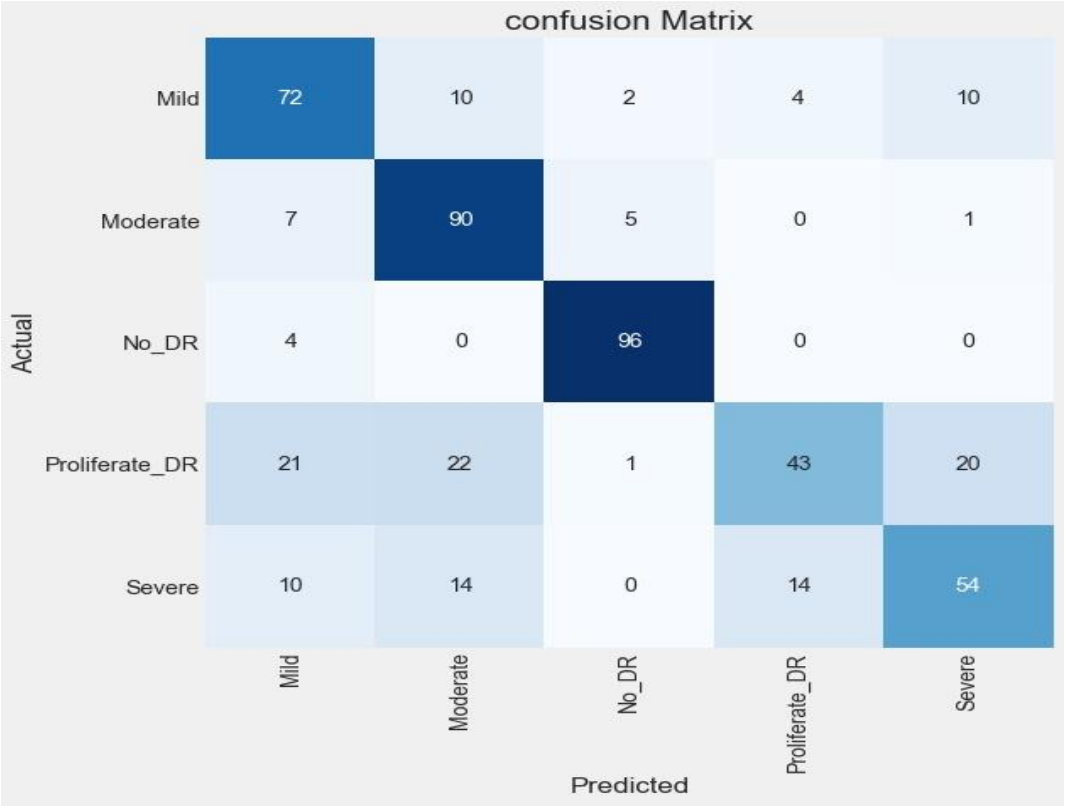


EfficientNetB0 loss and accuracy graph

Figure 5.2: Loss and accuracy graph for both the models



MobileNetv2 Confusion Matrix



EfficientNetB0 Confusion Matrix

Figure 5.3: Confusion matrix on the validation data.

	Precision	**Recall**	**f1-score**
No DR	0.93	0.91	0.92
Mild	0.44	0.41	0.43
Moderate	0.58	0.69	0.63
Severe	0.50	0.67	0.58
Proliferate DR	0.55	0.35	0.43
Accuracy	0.60		

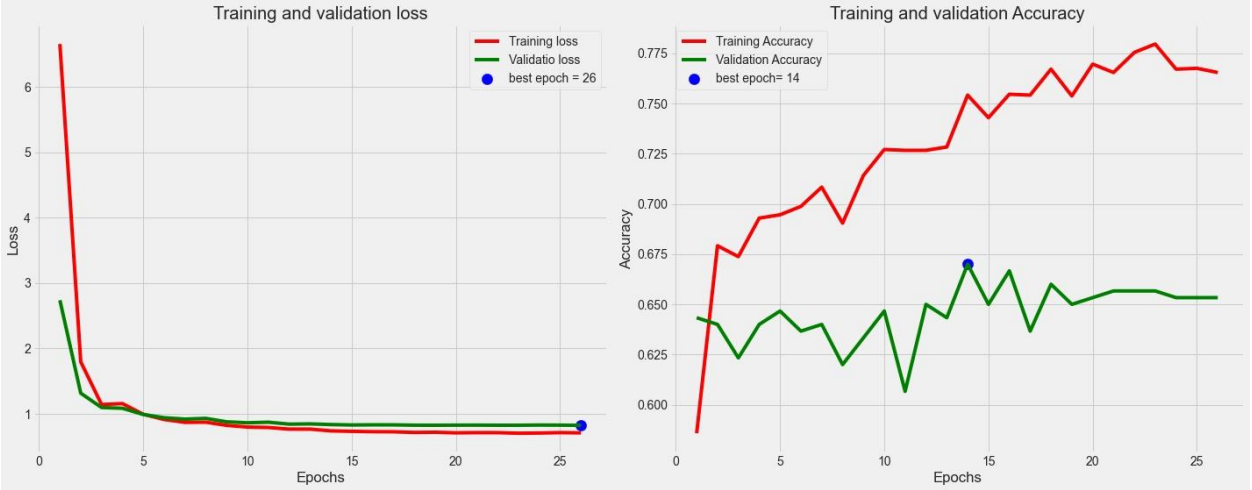
Table 9: Summary evaluation MobileNetV2

	Precision	**Recall**	**f1-score**
No DR	0.92	0.96	0.94
Mild	0.63	0.73	0.68
Moderate	0.66	0.87	0.75
Severe	0.64	0.59	0.61
Proliferate DR	0.70	0.40	0.51
Accuracy	0.71		

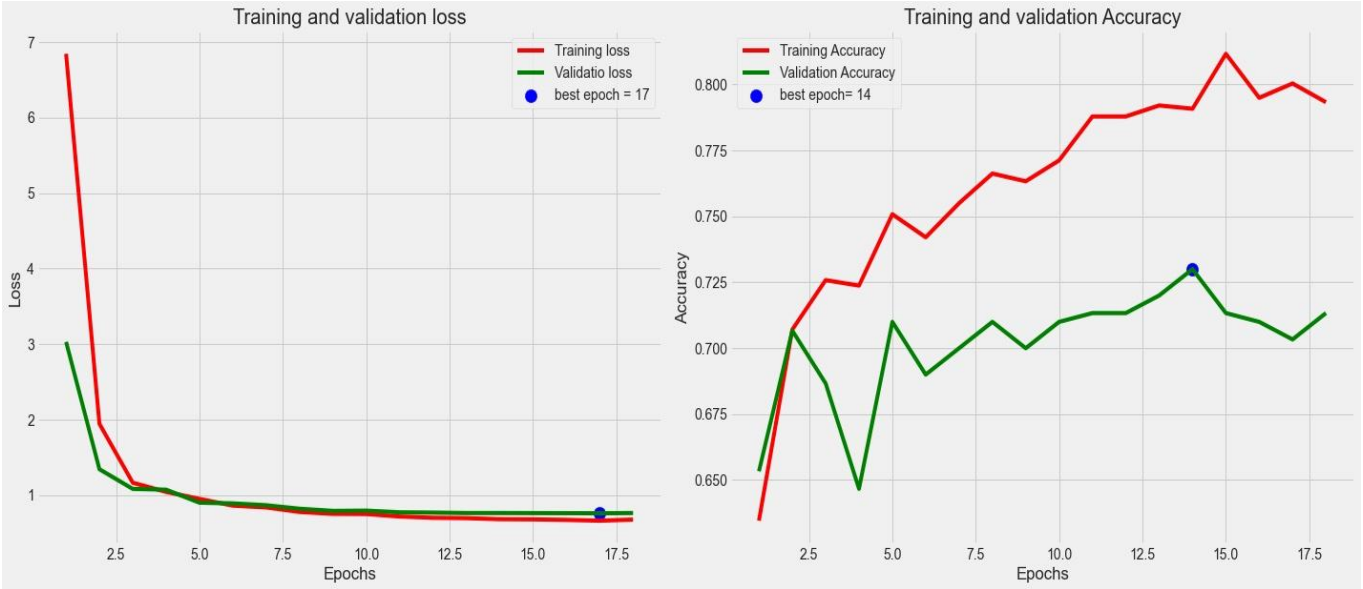
Table 10: Summary evaluation EfficientNetB0

5.3.2 Model trained with augmented dataset to classify 3 stages.

In this phase we tries to merge those close stages into one, that is mild and moderate as single stage and severe and Proliferate as another single stage, that is the model is generally trained to classify into 3 stages ,namely the No DR stage, Mild_Moderate DR stage, and Severe_Proliferate DR.Below is the detail graph, confusion matrix and evaluation table.



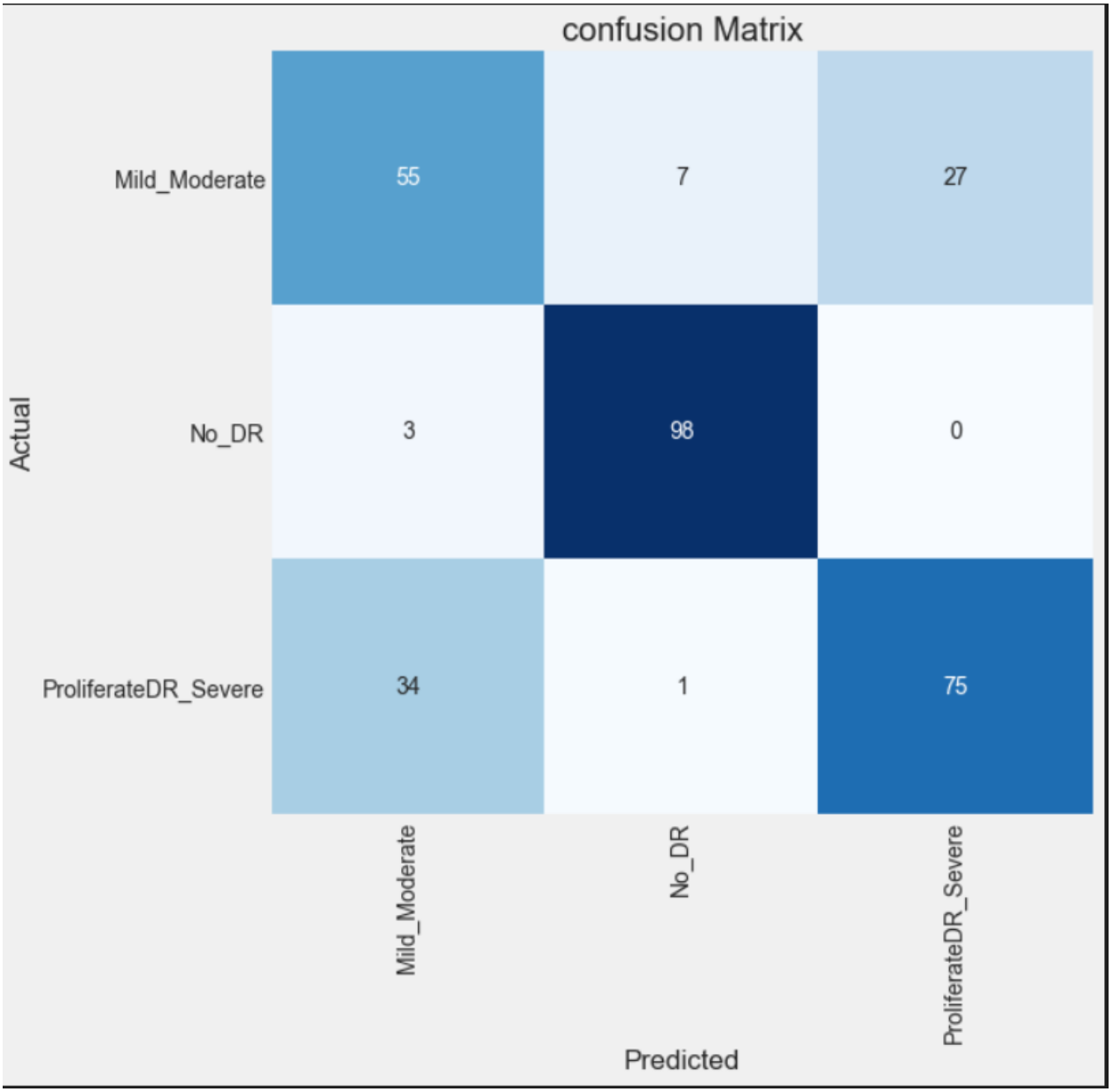
MobileNetV2 loss and accuracy graph



EfficientNetB0 loss and accuracy graph

Figure 5.4:Loss and accuracy graph for both models

MobileNetv2 Confusion Matrix



EfficientNetB0 Confusion Matrix

Figure 5.5:Confusion matrix for both the models on the validation data.

	Precision	Recall	f1-score
No DR	0.90	0.94	0.92
Mild_Moderate	0.57	0.56	0.57
Severe_Proliferate DR	0.66	0.65	0.65
Accuracy	0.72		

Table 11: Summary evaluation MobileNetV2

	Precision	Recall	f1-score
No DR	0.92	0.97	0.95
Mild_Moderate	0.60	0.62	0.61
Severe_Proliferate DR	0.74	0.68	0.71
Accuracy	0.76		

Table 12: Summary evaluation EfficientNetB0

5.3.3 Model trained with merged and preprocessed dataset

From the above and other experiments, we have absorbed several issues that affect the accuracy of the model. The first absorption is that making data augmentation affects those stages with less original data during the training and testing phases. So in order to avoid this, we try to merge the images from our two datasets to make a reasonable number of samples for each stage of DR. But since they have different types of formats, we preprocess twice and prepare the data for training. Another absorption is related to our two CNN models, in that it shows clearly that the EfficientNetB0 shows greater accuracy than MobilenetV2, and that the size of the final model is smaller in EfficientNetB0 also.

Figure 5.6: EfficientNetB0 Loss and Accuracy

Below is the graph, confusion matrix, and evaluation table for EfficientB0. From the confusion matrix, we can clearly see the expected difficulty, where the model has difficulties while trying to detect, especially on the last stages of DR.

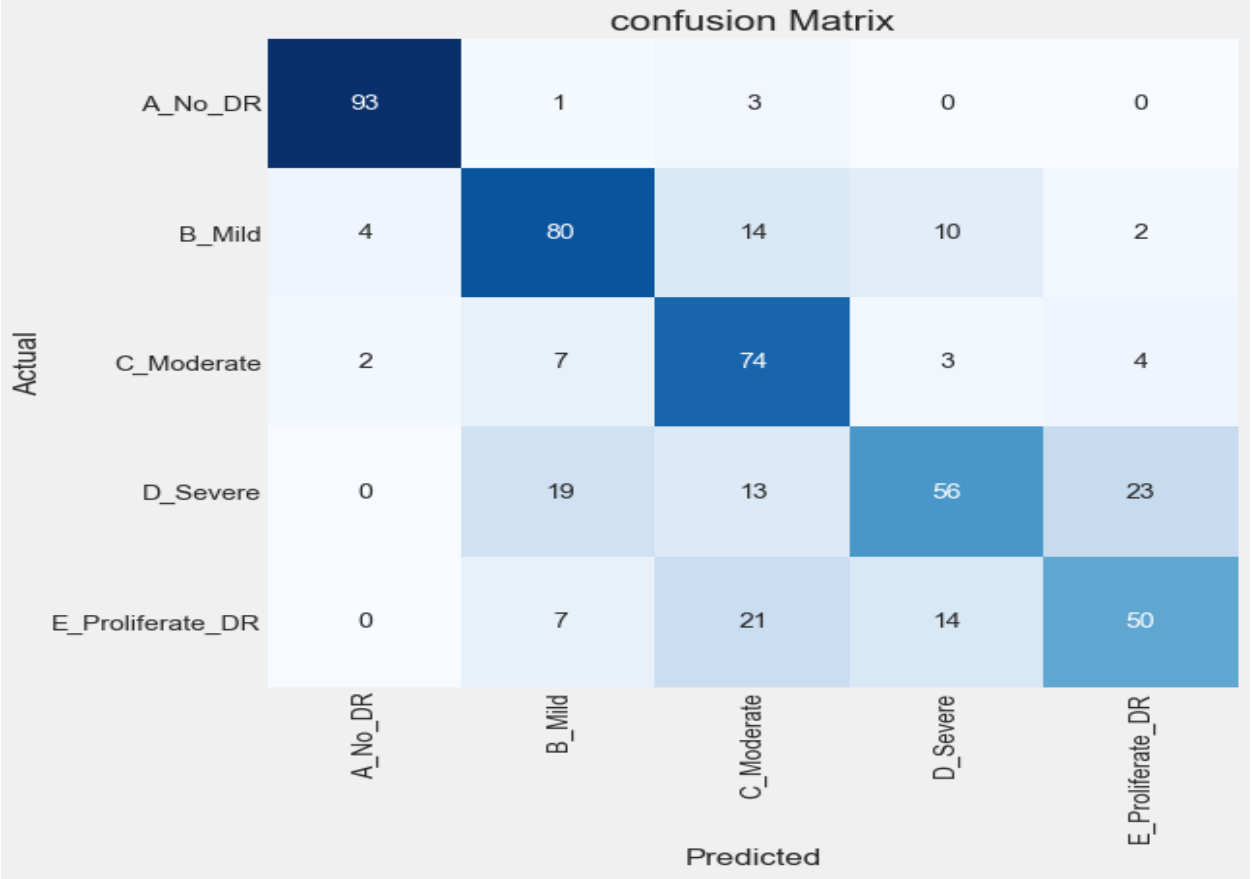


Figure 5.7: EfficientNetB0 confusion Matrix

	Precision	Recall	f1-score
No DR	0.94	0.96	0.95
Mild	0.70	0.73	0.71
Moderate	0.59	0.82	0.69
Severe	0.67	0.50	0.58
Proliferate DR	0.63	0.54	0.58
Accuracy	0.71		

Table 13: Summary evaluation EfficientNetB0

6. Concluding Discussion and Future Work

Below we have summed up the overall aim of the paper with a discussion on the findings and conclude with suggestions for things to be done in the future.

6.1 Conclusions

Diabetes is one of the most fast-growing diseases in recent times. According to various surveys, a patient having diabetes has around a 30% risk of getting Diabetic Retinopathy (DR). DR has different stages, ranging from mild to severe, and then finally PDR (Proliferative Diabetic Retinopathy). In the later stages of the disease, it leads to floaters, blurred vision, and eventually to blindness if not detected in the earlier stages. Although many computer vision-based techniques for automatic detection of DR using hands-on engineering and end-to-end learning approaches have been proposed there is still a need for easy access to such models.

The research in this paper is intended to help diabetic patients to remain cautious about their medical condition, by implementing a mobile-based DL detecting model for early analysis and detection of DR.

From the various already developed deep neural networks architectures we take a close look at MobileNetV2 and EfficientNetB0, since they are intended to be used to develop a model for mobile or embedded devices. To train and validate the network we use two Kaggle datasets. From our experiment and absorptions, it is crucial to pre-process the images before using them, and for that, we use Gaussian filter, the method that yields the best versions of the images which can be used to train and validate. With regard to DL network architecture, we have found that the EfficientNetB0 yields better accuracy than MobileNetV2, with a smaller-sized model. The model classifies the early stage as well as expected, which is the aim of the study, to help early detection before it reaches the final stage, although it has some difficulty classifying the last DR stages as shown in the confusion matrix in figure 10.

As explained earlier in the Related Work, it is hard to classify the DR in the early stages. However, our experiments with the suggested model show good results for the early stage classification, especially in separating the Non_DR

from the mild and moderate stages. Preprocessing of the images greatly helps to identify those early stages. But our experiment also displays less accuracy while trying to classify the last two stages. This may be due to the fact that the characteristic features of the later stages of development do not replace the features of the previous stages, but rather add new features to the existing. This means that the features of the earlier stages will still be visible in the later stages which could make the separation of these stages more difficult.

6.2 Discussion

Our aim in this project has been to make a small, manageable and efficient model, which could classify and separate images for all the 5 stages of DR. The desired small size of the model is intended to ensure that the tool can be used in mobile devices and not be dependent on cloud computing. A model for deep learning is trained to detect one or more characteristic features, which can be used to classify images based on those features. As we have already mentioned above, in DR the features are overlapping with the progression of the disease. This is a problem that needs to be addressed in more detailed studies. However, the model is still precise in detecting the presence of the disease in its earliest stages, compared to when it is not present at all. We also found that good pre-processing of the images is very important before training and validating the model. So the more thoroughly the images are pre-processed the better the accuracy of the model will be.
Our final model, based on the architecture of EfficientNetB0, has a size of 20.8 Mb. This is very small in comparison with many ordinary Deep neural networks which propabably have a size above 200Mb mostly. Despite this limited size it still shows a good classification accuracy, and especially with regards to the images with and without DR, as well as with those early stages of DR, i.e Mild and Moderate.

The unbalanced Kaggle dataset, and especially the small number of samples for the last stage, with the high complexity of features in the images, make the classification harder for the last stages.

However, we can still conclude that the current results already indicate the possibility of developing an interesting tool as a mobile application, for the

direct assessment of the degree of DR already from the initial phases, which today is a difficult task.

6.3 Future Work

As already mentioned, the model does not have great accuracy on the last stages. Although this could be partly due to the relatively small number of pictures for training and validation, the question still remains about why some classes are easier to separate than some of the others. This seems to be a systematic deviance, and therefore needs more investigation, possibly with a larger image pool. Thus, it would be very interesting to pursue the following paths forward:

1. Accessing more data of a higher quality.
2. Balancing the numbers of images for each category, to achieve a more balanced datasets on each subcategory of DR.
3. Improving the preprocessing techniques in order to increase the clarity of the images. This needs to be balanced carefully, so that the data is not destroyed through these transfers.
4. Increasing the overall accuracy for all the stages.
5. Automatically self retraining the model after deployment.
6. Taking a direct image from the mobile camera instead of using the funds' image, maybe with the help of an external lens or other magnifying devices.

Citations

1. W. R. Memon, B. Lal and A. A. Sahto, "Diabetic retinopathy", *The Prof. Med. J.*, vol. 24, pp. 234-238, 2017.

2. Qummar, Sehrish, et al. "A deep learning ensemble approach for diabetic retinopathy detection." *IEEE Access* 7 (2019): 150530-150539..

3. S. Jan, I. Ahmad, S. Karim, Z. Hussain, M. Rehman and M. A. Shah, "Status of diabetic retinopathy and its presentation patterns in diabetics at ophthalomogy clinics", *J. Postgraduate Med. Inst. (Peshawar-Pakistan)*, vol. 32, no. 1, pp. 24-27, Mar. 2018.

4. J. Amin, M. Sharif, M. Yasmin, H. Ali and S. L. Fernandes, "A method for the detection and classification of diabetic retinopathy using structural predictors of bright lesions", *J. Comput. Sci.*, vol. 19, pp. 153-164, Mar. 2017.

5. Priya, R., and P. Aruna. "Diagnosis of diabetic retinopathy using machine learning techniques." *ICTACT Journal on soft computing* 3.4 (2013)

6. R. Gargeya and T. Leng, "Automated identification of diabetic retinopathy using deep learning", *Ophthalmology*, vol. 124, no. 7, pp. 962-969, 2017.

7. N. Kashyap, D. K. Singh and G. K. Singh, "Mobile phone based diabetic retinopathy detection system using ANN-DWT," 2017 4th IEEE Uttar Pradesh

8. Xu, Kele, Dawei Feng, and Haibo Mi. "Deep convolutional neural network-based early automated detection of diabetic retinopathy using fundus image." *Molecules* 22.12 (2017): 2054.

9. Chakrabarty, Navoneel. "A deep learning method for the detection of diabetic retinopathy." *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. IEEE, 2018.

10. Rachel CAI ,'Study of Convolutional Neural Networks for Early Detection of Diabetic Retinopathy',2020

11.Kandel, Ibrahem, and Mauro Castelli. "Transfer learning with convolutional neural networks for diabetic retinopathy image classification. A review." *Applied Sciences* 10.6 (2020): 2021.

12. Sarki, Rubina, et al. "Convolutional neural networks for mild diabetic retinopathy detection: an experimental study." *bioRxiv*(2019): 763136.

13. Patel, Rupa, and Anita Chaware. "Quantizing MobileNet Models for Classification Problem." *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2021.

14. Pat, el, Sanskruti. "Diabetic retinopathy detection and classification using pre-trained convolutional neural networks." *International Journal on Emerging Technologies* 11.3 (2020): 1082-1087.

15. Lim, Gilbert, et al. "Different fundus imaging modalities and technical factors in AI screening for diabetic retinopathy: a review." *Eye and Vision* 7.1 (2020): 1-13

16. https://towardsdatascience.com/pruning-deep-neural-network-56cae1ec5505

[html]

17. https://zihuaweng.github.io/2018/08/14/mobilenet_xception/ [html]

18. Sifre, Laurent, and Prof Stéphane Mallat. "Rigid-Motion Scattering For Image Classification Author." *English. Supervisor: Prof. Stéphane Mallat. Ph. D. Thesis. Ecole Polytechnique* (2014).

19. Wang, Xiaoliang & Lu, Yongjin & Wang, Yujuan & Chen, Wei-Bang. (2018). Diabetic Retinopathy Stage Classification Using Convolutional Neural Networks. 465-471. 10.1109/IRI.2018.00074. [Image]

20. ”APTOS : Eye Preprocessing in Diabetic Retinopathy,” Kaggle, https://www.kaggle.com/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy, 2019, [Dataset].

21. Diabetic Retinopathy (resized), Resized version of the Diabetic Retinopathy Kaggle competition dataset https://www.kaggle.com/tanlikesmath/diabetic-retinopathy-resized,2019,[Dataset]

22. Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

23. Tan, Mingxing and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” *ArXiv* abs/1905.11946 (2019): n. Pag.

24. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprintarXiv:1409.1556, 2014.

25. Tajbakhsh N, Shin JY, Gurudu SR, Hurst RT, Kendall CB, Gotway MB, et al. Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans Med Imaging. 2016;35(5):1299–312.

26. Baheti, Bhakti, et al. "Eff-unet: A novel architecture for semantic segmentation in an unstructured environment." *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020.

27. Wang, Xiaoliang, et al. "Diabetic retinopathy stage classification using convolutional neural networks." *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, 2018.

28. Graham, Ben. "Kaggle diabetic retinopathy detection competition report." *University of Warwick* (2015): 24-26.