IDENTIFICATION OF DIABETIC RETINOPATHY SIGNS USING DEEP LEARNING

A Project Report Submitted by

Amartya Kalapahar (4NM16CS016)

Ankit Pandita (4NM16CS020)

Pratheek Shenoy K (4NM16CS217)

UNDER THE GUIDANCE OF

Dr. Jyothi Shetty
Professor

Department of Computer Science and Engineering

In partial fulfillment of the requirements for the award of the Degree of

Bachelor of Engineering in Computer Science & Engineering

From

Visvesvaraya Technological University, Belagavi

NITTE EDUCATION TRUST

N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
08258 - 281039 – 281263, Fax: 08258 – 281265

B.E.CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

April 2020

NITTE EDUCATION TRUST
N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
☎: 08258 - 281039 – 281263, Fax: 08258 – 281265
Department of Computer Science and Engineering
B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

CERTIFICATE

Certified that the project work entitled

"Identification of diabetic retinopathy signs using deep learning"

is a bonafide work carried out by

Amartya Kalapahar(4NM16CS016) *Ankit Pandita(4NM16CS020)*
Pratheeek Shenoy K(4NM16CS217)

in partial fulfillment of the requirements for the award of ***Bachelor of Engineering***

Degree in Computer Science and Engineering

prescribed by ***Visvesvaraya Technological University,***
Belagavi *during the year 2019-2020.*

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide **Signature of HOD** **Signature of Principal**

Semester End Viva Voce Examination

Name of the Examiners **Signature with Date**

1. _____

2. _____

ACKNOWLEDGEMENT

We believe that our project will be complete only after we thank the people who have contributed to making this project successful.

First and foremost, our sincere thanks to our beloved principal, **Dr. Niranjan N. Chiplunkar** for allowing us to carry out our project work at our college and providing us with all the needed facilities.

We express our deep sense of gratitude and indebtedness to our guide **Dr. Jyothi Shetty,** Professor, Department of Computer Science and Engineering, for her inspiring guidance, constant encouragement, support and suggestions for improvement during the course for our project.

We express our gratitude to the project co-ordinators **Mrs. Pallavi K N, Mrs. Asmita Poojary, Mr. Ranjan Kumar HS** and **Mr. Raju K.** for their constant support.

We also thank all those who have supported us throughout the entire duration of our project.

Finally, we thank the staff members of the Department of Computer Science and Engineering and all our friends for their honest opinions and suggestions throughout our project.

Amartya Kalapahar

Ankit Pandita

Pratheek Shenoy K

ABSTRACT

Diabetic Retinopathy (DR) is an eye disease associated with chronic diabetes. DR is the leading cause of blindness among working-age adults around the world and estimated it may affect more than 93 million people. Progression to vision impairment can be slowed or controlled if DR is detected in time, however, this can be difficult as the disease often shows few symptoms until it is too late to provide effective treatment. Currently, detecting DR is a time-consuming and manual process, which requires an ophthalmologist or trained clinician to examine and evaluate digital color fundus photographs of the retina, to identify DR by the presence of lesions associated with the vascular abnormalities caused by the disease.

The importance of diabetic retinopathy screening programs and difficulty in achieving reliable early diagnosis of diabetic retinopathy at a reasonable cost needs attention to develop a computer-aided diagnosis tool. Computer-aided disease diagnosis in retinal image analysis could ease mass screening of the population with diabetes mellitus and help clinicians in utilizing their time more efficiently. The recent technological advances in computing power, communication systems, and machine learning techniques provide opportunities for biomedical engineers and computer scientists to meet the requirements of clinical practice.

To overcome the above-said challenges in our project we developing a deep learning model that will classify the retinal images into the various stages of the diabetic retinopathy. In this project, we have implemented various deep learning architectures and compared their architectures and found a suitable architecture to solve the issue.

TABLE OF CONTENTS

Contents	Page
Title Page	i
Certificate	ii
Acknowledgement	iii
Abstract	iv
Table of contents	v
List of figures	viii
CHAPTER 1 INTRODUCTION	1-5
1.1 Overview	2
1.2 Problem Statement	2
1.3 Study Area	2
1.4 Objective	3
1.5 Motivation	4
1.6 Organization of the Report	4-5
CHAPTER 2 LITERATURE SURVEY	6-8
2.1 Related Work	6-8
CHAPTER 3 SYSTEM ANALYSIS & REQUIREMENTS	9-10
3.1 Relevance of Programming Language	9
3.2 Functional requirements	9-10
3.2.1 Software Requirements	9
3.2.2 Hardware Requirements	10

CHAPTER 4 SOFTWARE APPROACH 11-14
4.1 Dataset 11
4.2 Dataset Balancing 11
4.3 Data Preprocessing 11-12
4.4 Data Augmentation 12
4.5 Hyperparametrs 12-14

CHAPTER 5 SYSTEM IMPLEMENTATION 15-35
5.1 Data Preprocessing and Augmentation 15
5.2 Convolutional Neural Networks 15
5.3 Base Model 15-16
5.4 EficientNet Model 16-23
5.5 DenseNet Model 23-34
5.6 CapsuleNet Model 34-35

CHAPTER 6 RESULTS 36

CHAPTER 7 SCREENSHOTS 37-44

7.1 Sample Dataset 37

7.2 Dataset Visualization 39

7.3 Retinal Images after Prepossessing and Augmentation 39

7.4 Base Model 40

7.5 EfficientNet Model 41

7.6 DensNet Model 42

7.7 DenseNet Model Output Graph 42

7.8 CapsuleNet Model 43-44

CHAPTER 8 CONCLUSION AND FUTURE WORK 45

8.1 Conclusion 45

8.2 Future Work 45

REFERENCES 46-47

LIST OF FIGURES

Figure 2.1 Model by Gardner and others 7

Figure 2.2 Model by Nayak and others 8

Figure 2.3 Model by Acharya and others 8

Figure 4.1 Data Preprocessing Steps 12

Figure 5.1 Base Model Architecture 16

Figure 5.2 EfficientNet Model Architecture 17

Figure 5.3 DenseNet Model Architecture 24

Figure 5.4 CapsuleNet Model Architecture 35

Figure 7.1 Sample Dataset 37

Figure 7.2 Distribution of Dataset 38

Figure 7.3 Graphical Distribution of Dataset 38

Figure 7.4 Balanced Dataset Graph 39

Figure 7.5 Dataset after Preprocessing and Augmentation 39

Figure 7.6 Base Model Summary 40

Figure 7.7 EfficientNet Model Summary 41

Figure 7.8 DenseNet Model Summary 42

Figure 7.4 DenseNet Accuracy v/s Validation Accuracy Graph 42

Figure 7.5 CapsuleNet Model Summary 43-44

CHAPTER 1

INTRODUCTION

Diabetic retinopathy (DR), also known as diabetic eye disease, is when damage occurs to the retina due to diabetes. It can eventually lead to blindness. It is an ocular manifestation of diabetes. Despite these intimidating statistics, research indicates that at least 90% of these new cases could be reduced if there were proper and vigilant treatment and monitoring of the eyes. The longer a person has diabetes, the higher his or her chances of developing diabetic retinopathy. Diabetic retinopathy can be diagnosed into 5 stages: mild, moderate, severe, proliferative or no disease. The various signs and markers of diabetic retinopathy include microaneurysms, leaking blood vessels, retinal swellings, growth of abnormal new blood vessels and damaged nerve tissues.

DR detection is challenging because by the time human readers submit their reviews, often a day or two later, the delayed results lead to lost follow up, miscommunication, and delayed treatment. Clinicians can identify DR by the presence of lesions associated with the vascular abnormalities caused by the disease. While this approach is effective, its resource demands are high. The expertise and equipment required are often lacking in areas where the rate of diabetes in local populations is high and DR detection is most needed. The need for a comprehensive and automated method of DR screening has long been recognized, and previous efforts have made good progress using image classification, pattern recognition, and machine learning.

The current research in diagnosing diabetic retinopathy has been based on explicit extraction of features like microaneurysms and lesions through which the classification is performed. There has also been researching using machine learning techniques to classify the image as normal or diseased. Our project aims at proposing a diabetic retinopathy diagnosis model that automatically learns features that are pivotal in diagnosing the stage of the disease without explicit or manual feature extraction.

1.1 OVERVIEW

Diabetic retinopathy (DR), also known as diabetic eye disease, is a medical condition in which damage occurs to the retina due to diabetes mellitus. It is a leading cause of blindness. Diabetic Retinopathy is one of the major causes of blindness in the present world. Diabetic retinopathy affects up to 80 percent of those who have had diabetes for 20 years or more. If detected early enough, effective treatment of DR is available, making this a vital process. Significant work has been done on detecting the features of DR using automated methods such as support vector machines and k-NN classifiers. The majority of these classification techniques are on two-class classification for DR or no DR. Convolutional Neural Networks (CNNs), a branch of deep learning, have an impressive record for applications in image analysis and interpretation, including medical imaging.

1.2 PROBLEM STATEMENT

Diabetic retinopathy (DR), also known as diabetic eye disease, is when damage occurs to the retina due to diabetes. It can eventually lead to blindness. It is an ocular manifestation of diabetes. Despite these intimidating statistics, research indicates that at least 90% of these new cases could be reduced if there were proper and vigilant treatment and monitoring of the eyes. DR detection is challenging because by the time human readers submit their reviews, often a day or two later, the delayed results lead to lost follow up, miscommunication, and delayed treatment. Clinicians can identify DR by the presence of lesions associated with the vascular abnormalities caused by the disease. While this approach is effective, its resource demands are high.

The project mainly aims at providing a diabetic retinopathy diagnosis model that can classify the retinal image into 5 different stages of DR.

.

1.3 STUDY AREA

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. It is comprised of one or more convolutional layers (often with subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D hierarchical structure of an input image. This is achieved with local connections and tied weights followed by some form of pooling which results in translation-invariant features. CNN also considers the hierarchical representation of images while training by stacking multiple trainable stages on each other.

.

1.4 OBJECTIVE

In this project, we are using the Kaggle dataset. The dataset retinal images are preprocessed using data augmentation techniques. By using the Deep learning algorithms, we are extracting the features from the retinal images and Classifying them into 5 different stages of Diabetic Retinopathy.

- Pre-processing of the Retinal images to get a clear set of images.
- Balance the available dataset to get a uniform number of images in each class.
- Classify the preprocessed retinal images into the various classes.
- Implement various Deep Learning models which are suitable for the dataset.
- Calculate and Compare the efficiency, Accuracy of various models.

1.5 MOTIVATION

Diabetic Retinopathy is one of the major causes of blindness in the present world. Diabetic retinopathy affects up to 80 percent of those who have had diabetes for 20 years or more. If detected early enough, effective treatment of DR is available, making this a vital process. Hence we intend to develop a deep learning model that detects the presence of DR, classifies the DR into 5 levels of DR.

1.6 ORGANIZATION OF THE CHAPTERS

The project report has been organized under nine chapters, which are as follows:

Chapter I: Introduces the main idea of the project. It gives a brief knowledge of the aim and methodology of the same.

Chapter II: It includes literature survey of related works.

Chapter III: Discusses the system requirements that are needed for the project.

Chapter IV: Includes the application details.

Chapter V: Includes the implementation details of the project, the application is explained in detail.

Chapter VI: Discuss the results of the project.

Chapter VII: Includes the screenshots of the application.

Chapter VIII: outlines conclusions and future work that can be done.

CHAPTER 2

LITERATURE SURVEY

2.1 Related Work

2.1.1 Automatic detection of diabetic retinopathy using an artificial neural network (1996) by Gardner and others:

147 diabetic and 32 normal images were captured from a fundus camera, stored on the computer, and analyzed using a backpropagation neural network. The network was trained to recognize features in the retinal image. The effects of digital filtering techniques and different network variables were assessed. 200 diabetic and 101 normal images were then randomized and used to evaluate the network's performance for the detection of diabetic retinopathy against an ophthalmologist. Detection rates for the recognition of vessels, exudates, and hemorrhages were 91.7%, 93.1%, and 73.8% respectively. When compared with the results of the ophthalmologist, the network achieved a sensitivity of 88.4% and a specificity of 83.5% for the detection of diabetic retinopathy.

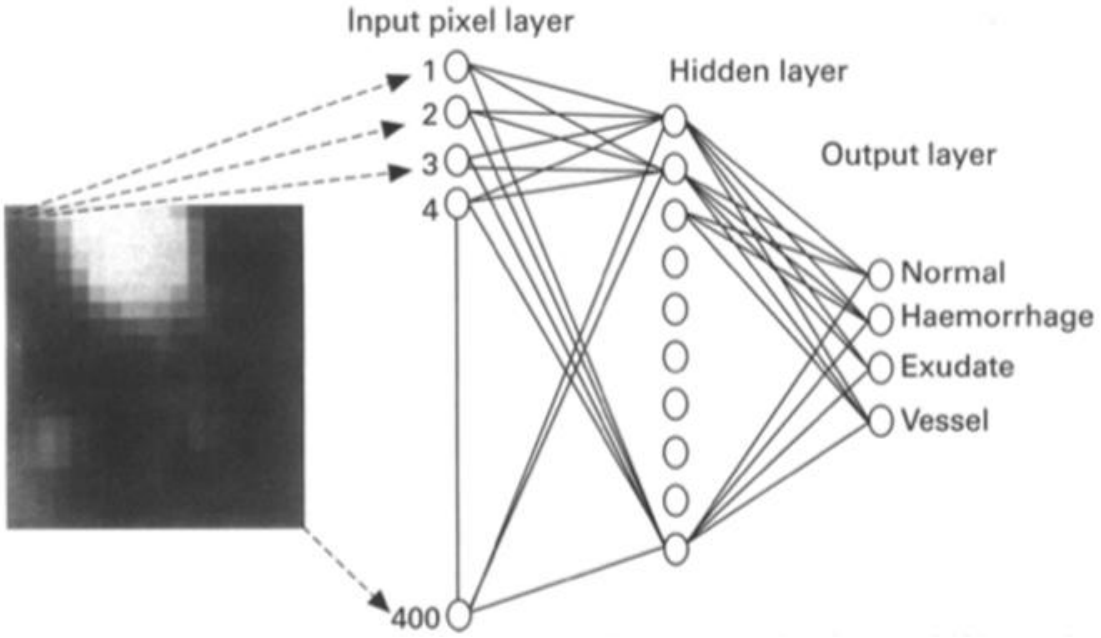


*Figure2.1***:** *Model by Gardner and others.*

2.1.2 Automatic detection of diabetic retinopathy using Digital Fundus Images (2009) by Nayak and others:

In this study, 331 fundus images were analyzed. Five groups of classification were done i.e normal retina, mild non-proliferative diabetic retinopathy, moderate non-proliferative diabetic retinopathy, severe non-proliferative diabetic retinopathy, and proliferative diabetic retinopathy. Four salient features: blood vessels, microaneurysms, exudates, and hemorrhages were extracted from the raw images using image-processing techniques and fed to the SVM for classification. A sensitivity of more than 82 % and specificity of 86 %.

feature extraction was required on all images in both training and testing which can be time-consuming.

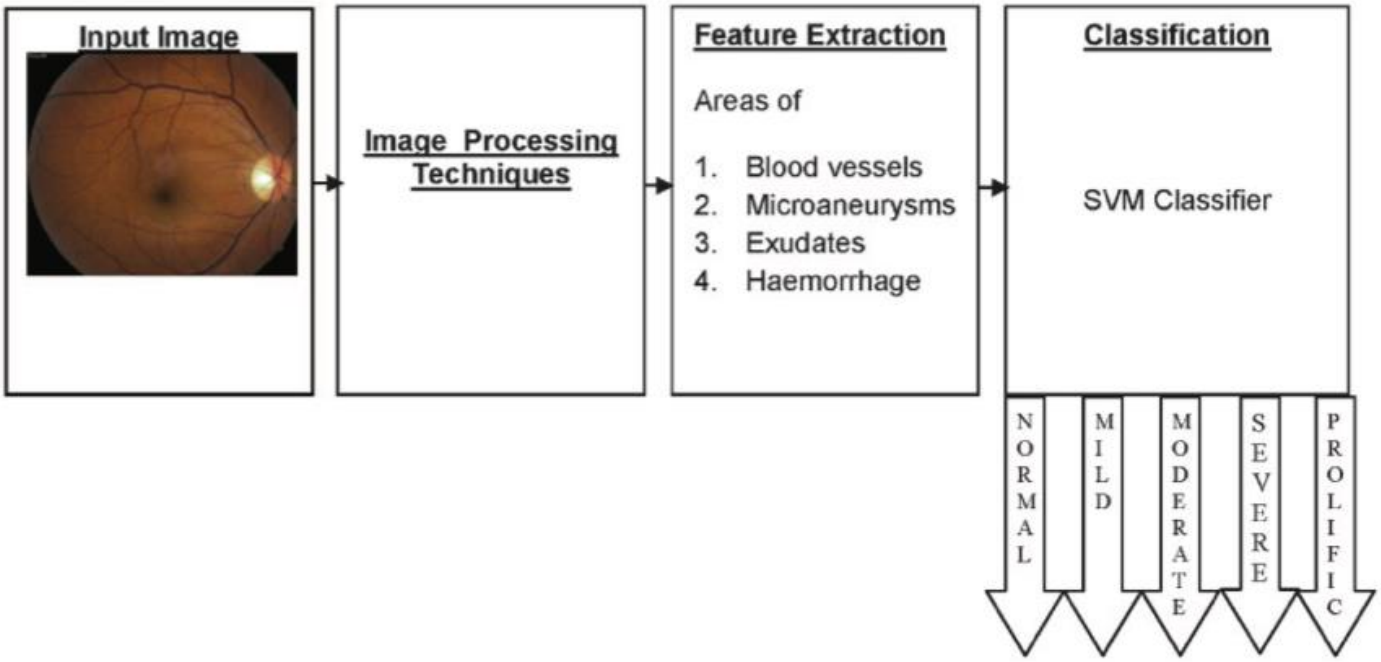


Figure 2.2: Model by Nayak and others

2.1.3 Computer-based detection of diabetic retinopathy stages using digital fundus images (2008) by Acharya and others:

In this study, five class classification that has been carried out has used SVM. The parameters are extracted from the raw images using the HOS(High Order Spectra) techniques and fed to the support vector machine (SVM) classifier. The sensitivity of 82% and specificity of 88% was achieved.

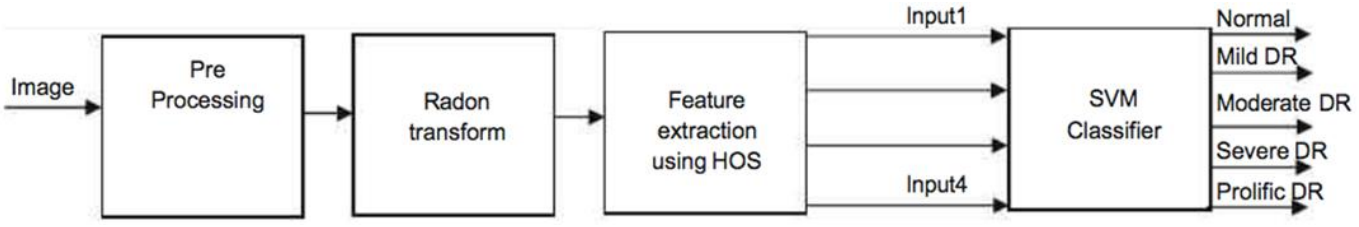


Figure 2.3: Model by Acharya and others

CHAPTER 3

SYSTEM ANALYSIS AND REQUIREMENTS

3.1 Relevance of Programming Language

Python is a general-purpose and high-level programming language. Created by Guido van Rossum and first released in 1991. Python is a powerful and object-oriented high-level programming language. It has very simple easy-to-use syntax. It works on cross-platform operating systems and can be used across to develop a wide range of applications including those intended for image processing, text processing, web, and enterprise-level using scientific, numeric and data from the network. The simple syntax rules of the programming language further make it easier for you to keep the code base readable and application maintainable. As an open-source programming language, Python helps you to curtail software development cost significantly. Python is famous for a rich selection of libraries, especially for data science. It's the root cause of why Python is considered the go-to solution for deep learning.

TensorFlow was originally developed by engineers and researchers at Google to meet their needs for a system that can build and train neural networks to find and decipher correlations and patterns. The process was designed as analogous to the ways humans reason and learn. The flexible, high-performance architecture of the open-source library makes it easy to deploy numerical computation across multiple platforms, as well as from desktops to server clusters to mobile devices.

3.2 FUNCTIONAL REQUIREMENTS

3.2.1 Software Requirements:

- Jupiter Notebook
- Docker
- TensorFlow
- Fastai
- Keras
- Pandas
- OpenCV
- Numpy
- PIL

3.2.2 Hardware Requirements:

- Operating system: Ubuntu preferred
- RAM: 8GB and above.
- Processor: I5 processor.
- UI: Terminal.

CHAPTER 4

SOFTWARE APPROACH

4.1 Dataset

We have used a resized version of the Diabetic Retinopathy Kaggle competition dataset collected from Aravind hospital. It contains 35,126 high-resolution retina images taken under a variety of imaging conditions. The Left and right fields are provided for every subject. Images are labeled with a subject id as well as either left or right (e.g. 1_left.jpeg is the left eye of patient id 1). It is collected in a CSV file with a class label for each image. Dataset is divided into 5 classes i.e Normal, Mild NPDR, Moderate NPDR, Severe NPDR, PDR.

4.2 Dataset Balancing

Imbalanced classes are a common problem in machine learning classification where there is a disproportionate ratio of observations in each class. Class imbalance can be found in many different areas including medical diagnosis, spam filtering, and fraud detection. The Problem with Imbalanced Classes is that most deep learning algorithms work best when the number of samples in each class are about equal. This is because most algorithms are designed to maximize accuracy and reduce errors.

Some methods for dealing with class imbalance are Oversample minority class, changing the Algorithm, changing the performance metric. Oversampling can be defined as adding more copies of the minority class.

4.3 Data Preprocessing

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

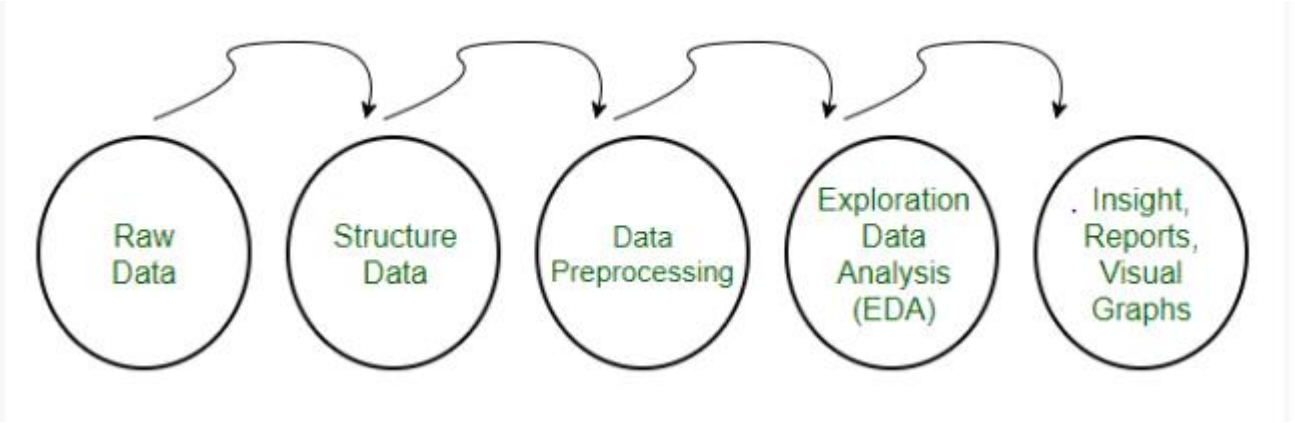


Figure4.1: Data Preprocessing steps

4.4 Data Augmentation

Data augmentation is the process of increasing the amount and diversity of data. We do not collect new data, rather we transform the already present data. Data augmentation is an integral process in deep learning, as in deep learning we need large amounts of data and in some cases, it is not feasible to collect thousands or millions of images, so data augmentation comes to the rescue. It helps us to increase the size of the dataset and introduce variability in the dataset.

The most commonly used operations are:

- Rotation: Rotation operation as the name suggests, just rotates the image by a certain specified degree.
- **Shearing**: Shearing is also used to transform the orientation of the image.
- Zooming: Zooming operation allows us to either zoom in or zoom out.
- Cropping: Allows us to crop the image or select a particular area from an image.
- Flipping: Allows us to flip the orientation of the image. We can use horizontal or vertical flip.
- Changing the brightness level.

4.5 Hyperparameters

Model Hyperparameters are the properties that govern the entire training process. Hyperparameters are important because they directly control the behavior of the training algorithm and have a significant impact on the performance of the model is being trained. Choosing appropriate hyperparameters plays a crucial role in the success of our neural network architecture. Since it makes a huge impact on the learned model.

Choosing good hyperparameters gives two benefits:

- Efficiently search the space of possible hyperparameters.
- Easy to manage a large set of experiments for hyperparameter tuning.

4.5.1 Optimizer

They tie together the loss function and model parameters by updating the model in response to the output of the loss function. In simpler terms, optimizers shape and mold your model into its most accurate possible form by futzing with the weights. The loss function is the guide to the terrain, telling the optimizer when it's moving in the right or wrong direction. In our project, we have used adam as the optimizer. Adam stands for adaptive moment estimation and is another way of using past gradients to calculate current gradients. Adam also utilizes the concept of momentum by adding fractions of previous gradients to the current one.

4.5.2 Learning Rate

The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.

4.5.3 Batch Size

Batch size is the total number of training samples present in a single min-batch. Choosing a batch size that is too small will introduce a high degree of variance (noisiness) within each batch as it is unlikely that a small sample is a good representation of the entire dataset. Conversely, if the batch size is too large, it may not fit in the memory of the compute instance used for training and it will tend to overfit the data. It's important to note that batch size is influenced by other hyperparameters such as the learning rate so the combination of these hyperparameters is as important as batch size itself.

4.5.4 Epochs

An epoch elapses when an entire dataset is passed forward and backward through the neural network exactly one time. If the entire dataset cannot be passed into the algorithm at once, it must be divided into mini-batches.

4.5.5 Loss Function

Data augmentation Loss function is used as a measurement of how good a prediction model does in terms of being able to predict the expected outcome. To keep a check on how accurate the solution is, loss functions are used. Loss functions used in our project are cross-entropy and mean squared error.

Cross-Entropy Loss: It measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. A perfect model would have a loss of 0.

Mean Squared Error: Mean square error is measured as the average of the squared difference between predictions and actual observations. It's only concerned with the average magnitude of error irrespective of their direction. However, due to squaring, predictions that are far away from actual values are penalized heavily in comparison to less deviated predictions.

4.5.6 Metric

Metrics are used to monitor and measure the performance of a model (during training, and test). Evaluation metrics are used to measure the quality of the statistical or machine learning model. Some of the metrics are

Confusion Matrix (error matrix): It is a tabular visualization of the model predictions versus the ground-truth labels. Each row of confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class.

Classification Accuracy: It is defined as the number of correct predictions divided by the total number of predictions, multiplied by 100. It is only suitable when there is an equal number of observations in each class and that all predictions and prediction errors are equally important.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 Dataset Preprocessing and Augmentation

The Kaggle dataset collected is highly imbalanced so various data preprocessing and augmentation techniques were used to balance the dataset.

The inputs were 512x512 images which were augmented in real-time by

- *Cropping* with certain probability
- *Color balance* adjustment
- *Brightness* adjustment
- *Contrast* adjustment
- *Flipping* images with 50% chance
- *Rotating* images by x degrees, with x an integer in [0, 360]
- *Zooming* (equal cropping on x and y dimensions)

5.2 Convolutional Neural Networks (CNNs)

After balancing the dataset, it is provided as input for various CNN Architectures to classify the retinal images into various classes. The results of these architectures are compared.

Some of the CNN Architectures used are:

- Base Model
- EfficientNet Model
- DenseNet Model
- CapsNet Model

5.3 Base Model

It is the common convolutional neural network architecture. CNN is now the go-to model on every image related problem. The main advantage of CNN is that it automatically detects the important features without any human supervision. CNN is also computationally efficient. It uses special convolution and pooling operations and performs parameter sharing. This enables CNN models to run on any device, making them universally attractive.

A CNN model can be thought of as a combination of two components: the feature extraction part and the classification part. The convolution and pooling layers perform feature extraction.

The main building block of CNN is the convolutional layer. Convolution is a mathematical operation to merge two sets of information. In our case, the convolution is applied to the input data using a convolution filter to produce a feature map.

After a convolution operation, we usually perform *pooling* to reduce the dimensionality. This enables us to reduce the number of parameters, which both shortens the training time and combats overfitting. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact.

Model Hyperparameters:

- Optimiser = Adam
- Learning rate = 0.0001
- Batch Size = 16
- Epochs = 50

Loss Function:

- Categorical Crossentropy

Metrics:

- Accuracy

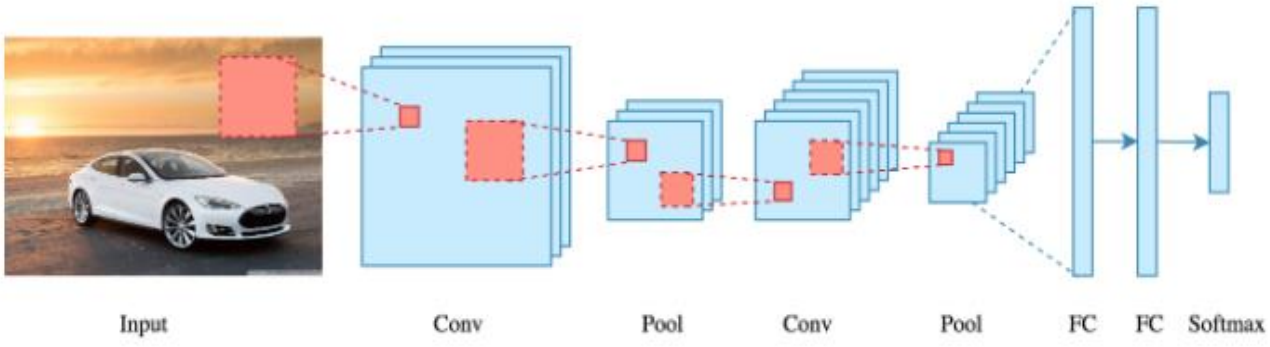


Figure5.1: Base Model Architecture

5.4 EfficientNet Model

EfficientNet model is an addition to the CNN model. Published by Google with an accuracy improvement of up to 6% on the ImageNet data. To scale up CNN, the depth of layers was increased by 20%, the width by 10% and the image resolution by 15% This compound scaling formula is used to scale up the EfficientNet from B0-B7 (increasing layers).

Hyperparameters:

- Optimizer = Adam

- Learning Rate = 0.0001
- Batch Size = 16
- Epochs = 30

Loss Function

- Kappa Loss
- Cross entropy

Metrics:

- Accuracy
- Quadratic Weighted Kappa

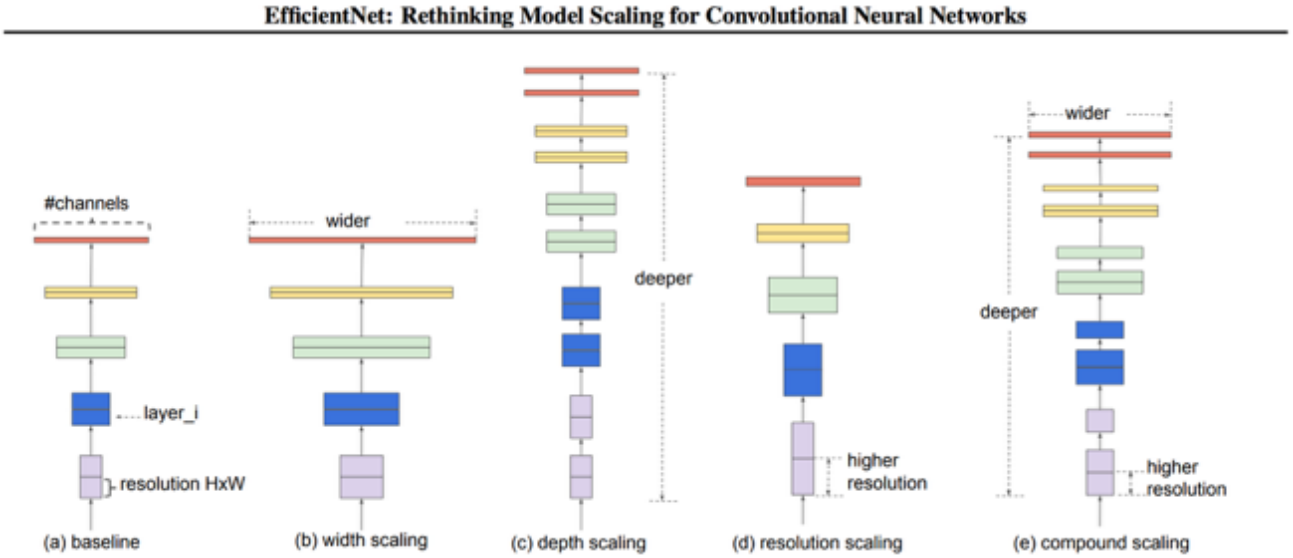


Figure 5.2: EfficientNet Model Architecture

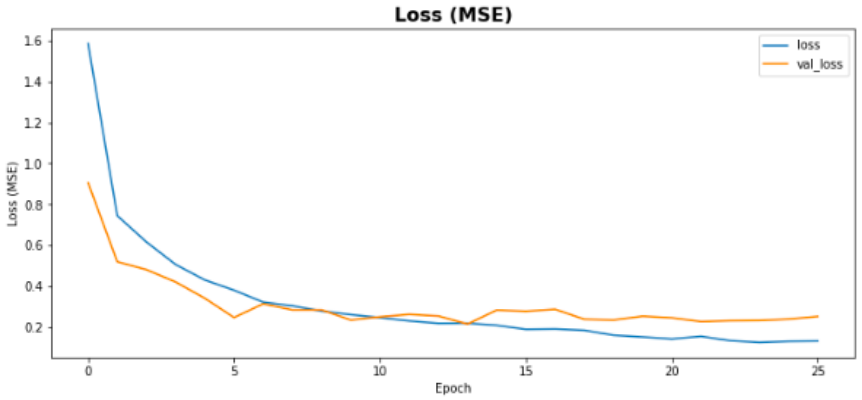
EfficientNet has various variants. In this project, we have implemented 2 variants of EfficientNet ie. EfficientNet-B5, EfficientNet-B6.

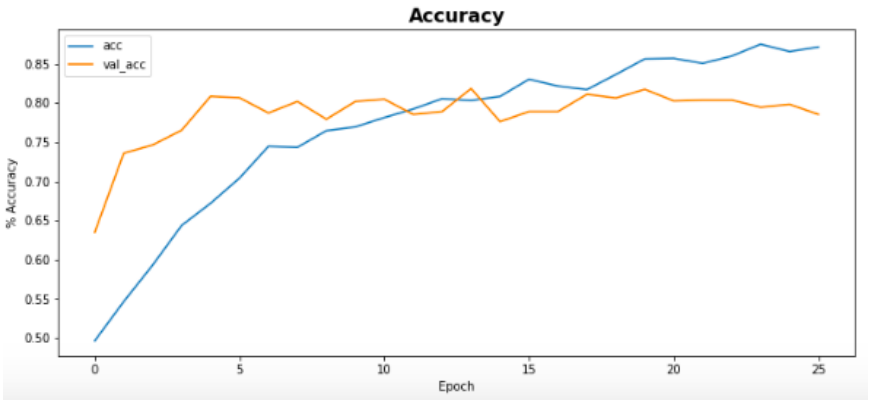
EfficientNet B5 Model

Architecture:	Output Shape	Parameters
EffNetB5	(None,15,15,2048)	28513520
GAP Layer	(None,2048)	0
Dropout Layer	(None,2048)	0
Dense Layer	(None,5)	10245
Output Layer	(None,1)	6

Total params: 28,523,771
Trainable params: 28,351,035
Non-trainable params: 172,736

Training Loss & Accuracy Visualization:





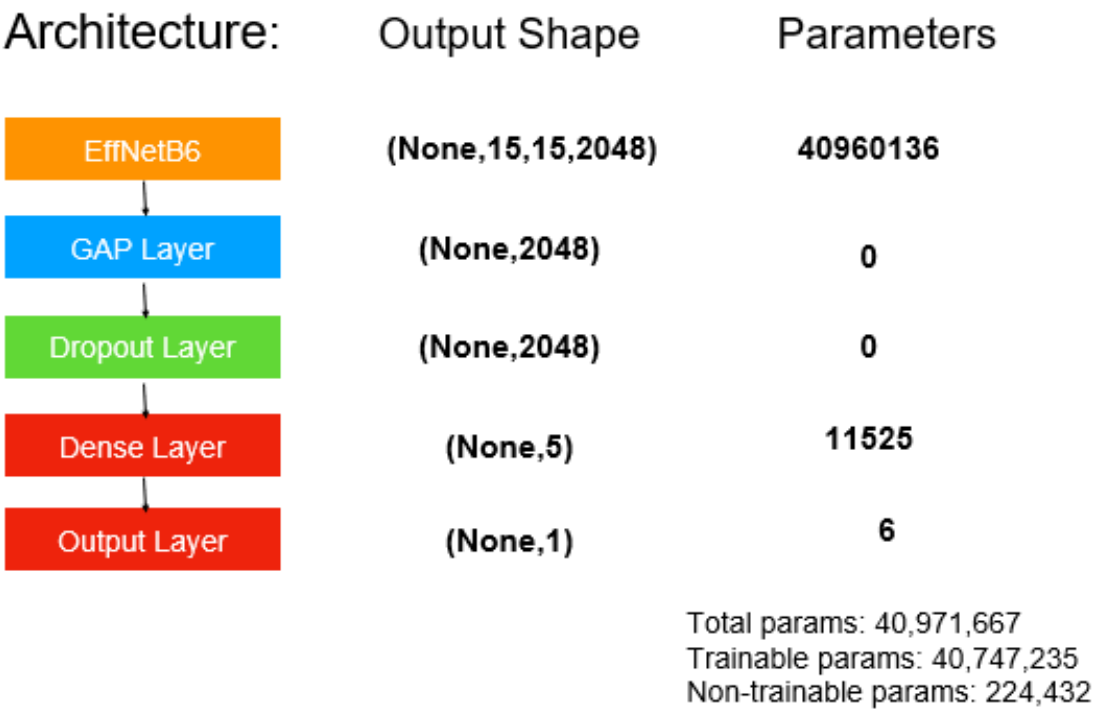
Confusion Matrix:

```
[[247   4   0   0   0]
 [  2  29  22   3   0]
 [  0  10 113  41   0]
 [  0   0   8  18   3]
 [  0   0   5  35   9]]
```

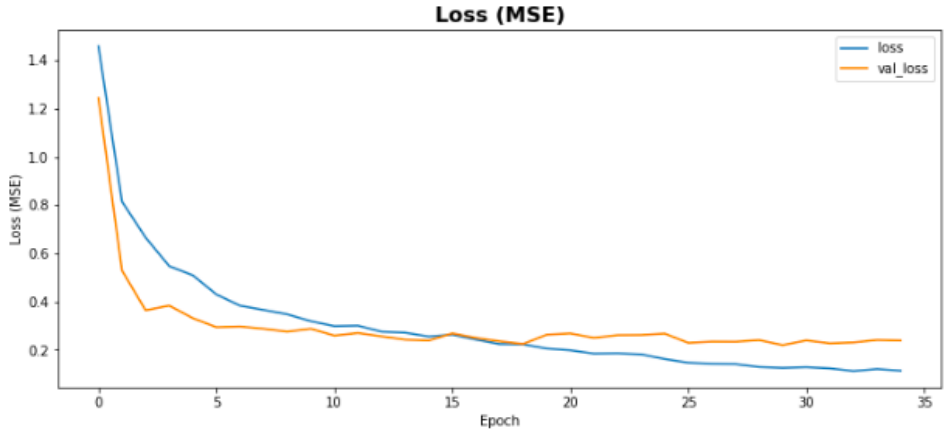
Classification Report:

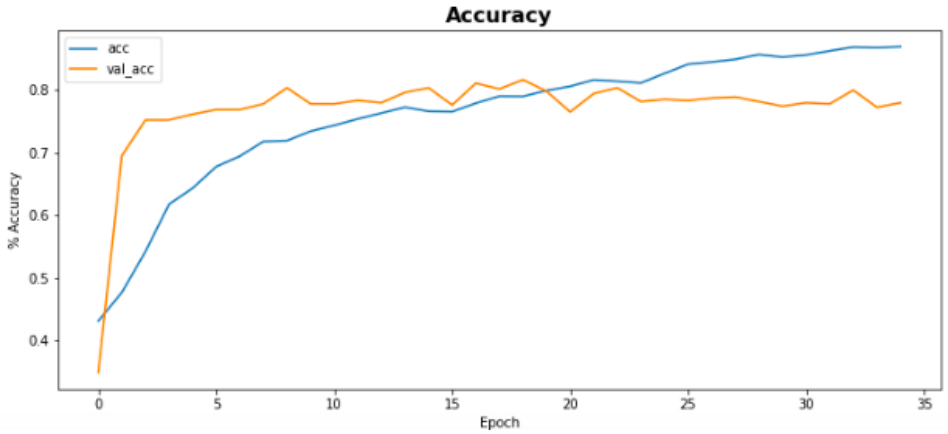
	precision	recall	f1-score	support
0	0.99	0.98	0.99	251
1	0.67	0.52	0.59	56
2	0.76	0.69	0.72	164
3	0.19	0.62	0.29	29
4	0.75	0.18	0.30	49
avg / total	0.83	0.76	0.77	549

EfficientNet B6 Model



Training Loss & Accuracy Visualization:





Confusion Matrix:

```
[[246   7   0   0   0]
 [  1  31  24   0   1]
 [  0  15 123  25   0]
 [  0   0  13  12   4]
 [  0   0   5  24  18]]
```

Classification Report:

	precision	recall	f1-score
	0.99	0.98	0.99
	0.67	0.52	0.59
	0.76	0.69	0.72
	0.19	0.62	0.29
	0.75	0.18	0.30
	0.83	0.76	0.77

Results from Initial Experiment On The Validation Set

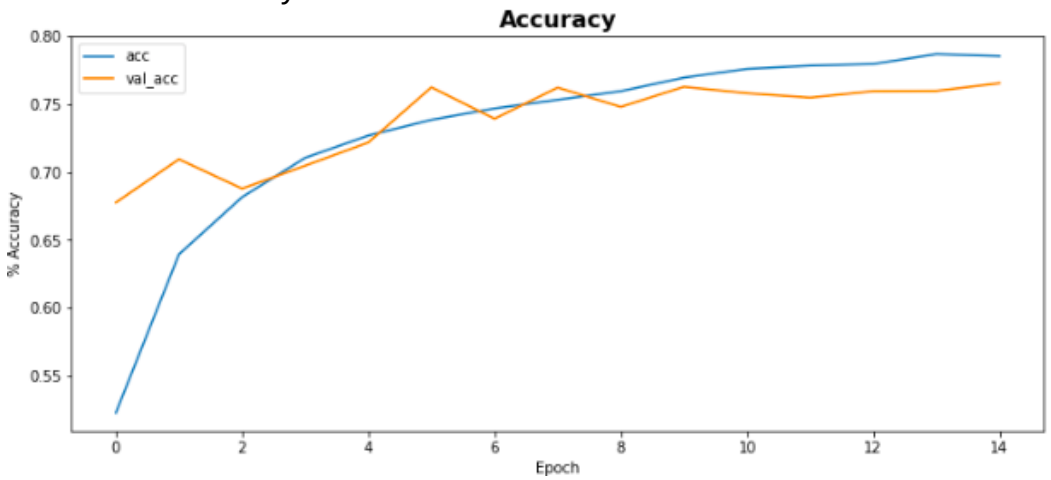
Models	EfficientNet B5	EfficientNet B6
Accuracy	0.7577	0.7832
Cohen Kappa Score	0.9195	0.92697

All the proposed models were trained using Aptos19(3.6k) dataset to find the best model among them. From the above table, we can see that EfficientNet-B6 gives the best result. The models were evaluated based on cohen kappa score and accuracy on the validation set.

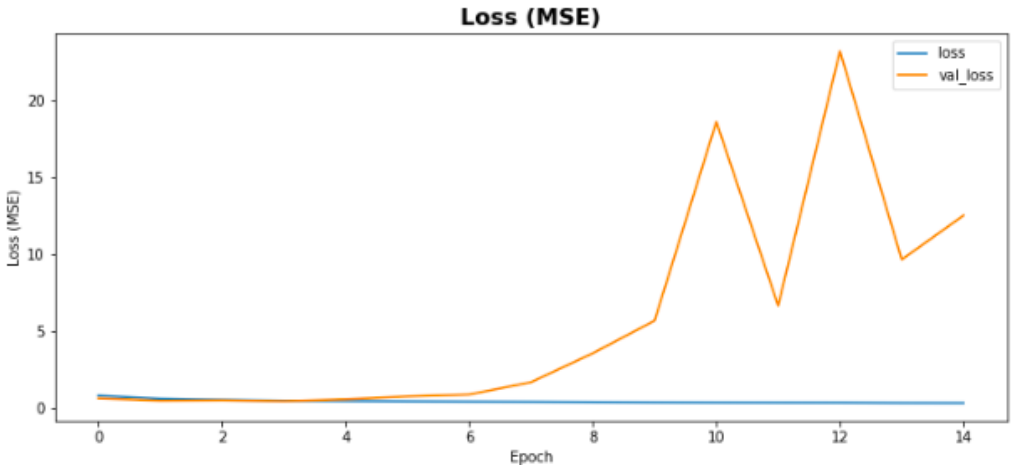
EfficientNet B6 with 2015 Dataset

The best performing model was again trained with the 2015 Diabetic Retinopathy dataset.

Train Vs. Validation Accuracy:



Train Vs. Validation Loss:



Confusion Matrix:

```
[[3488  333   37    5    6]
 [ 276   68   21    0    0]
 [ 230  197  272  120    6]
 [   4    7   21   73    9]
 [   1    2   13   31   48]]
```

Classification Report:

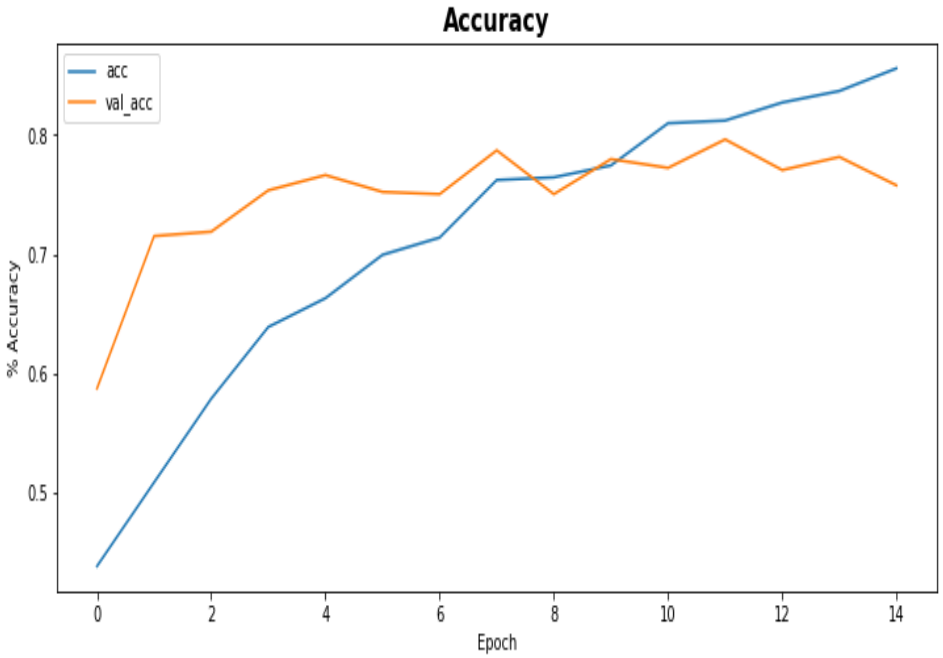
	precision	recall	f1-score	support
0	0.87	0.90	0.89	3869
1	0.11	0.19	0.14	365
2	0.75	0.33	0.46	825
3	0.32	0.64	0.43	114
4	0.70	0.51	0.59	95
total	0.78	0.75	0.75	5268

Accuracy: 0.7587

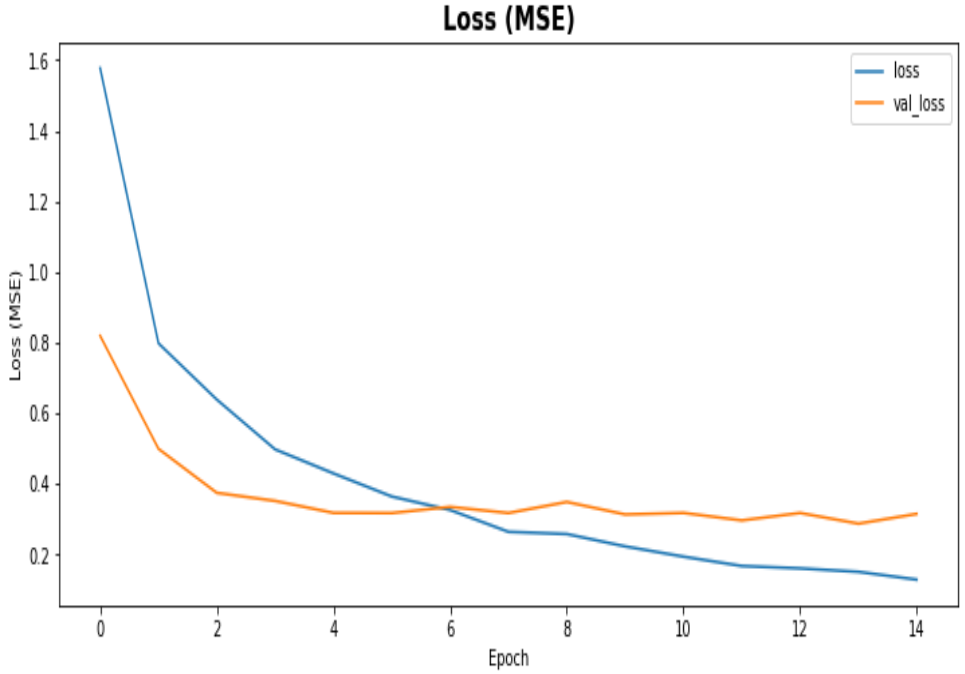
Cohen Kappa Score: 0.7593

EfficientNet B6 without Augmentation on Aptos

Train Vs. Validation Accuracy:



Train Vs. Validation Loss:



Confusion Matrix:

```
[[245   6   1   0   0]
 [  3  30  21   1   1]
 [  0  14 118  31   1]
 [  0   0   4  18   7]
 [  0   1  11  15  21]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.97	0.98	252
1	0.59	0.54	0.56	56
2	0.76	0.72	0.74	164
3	0.28	0.62	0.38	29
4	0.70	0.44	0.54	48
avg / total	0.82	0.79	0.80	549

Accuracy : 0.7868852459016393

Cohen Kappa Score is: 0.90511

5.5 DenseNet Model

DenseNet stands for Densely Connected Convolutional Networks. It was introduced by Gao Huang et al. DenseNet is a network architecture where each layer is directly connected to every other layer in a feed-forward fashion. For each layer, the feature maps of all preceding

layers are treated as separate inputs whereas its own feature maps are passed on as inputs to all subsequent layers.

A DenseNet consists of dense blocks. Each dense block consists of convolution layers. After a dense block, a transition layer is added to proceed to the next dense block. Every layer in a dense block is directly connected to all its subsequent layers. Consequently, each layer receives the feature-maps of all preceding layers.

Each convolution layer is consists of three consecutive operations: batch normalization (BN), followed by a rectified linear unit (ReLU), and a 3 × 3 convolution (Conv). An essential part of convolutional networks is down-sampling layers that change the size of feature-maps. To facilitate down-sampling in DenseNet architecture it divides the network into multiple densely connected dense blocks.

The layers between blocks are transition layers, which do convolution and pooling. The transition layers consist of a batch normalization layer and a 1×1 convolutional layer followed by a 2×2 average pooling layer.

In DenseNet, it introduces connections from one layer to all its subsequent layers in a feed-forward fashion. This connection is done using concatenation, not through summation. Each layer is receiving a “collective Knowledge” from all preceding layers.

Hyperparameters:

- Optimizer = Adam
- Learning Rate = 0.0001
- Batch Size = 32
- Epochs = 15

Loss Function

- Binary Cross entropy

Metrics:

- Accuracy

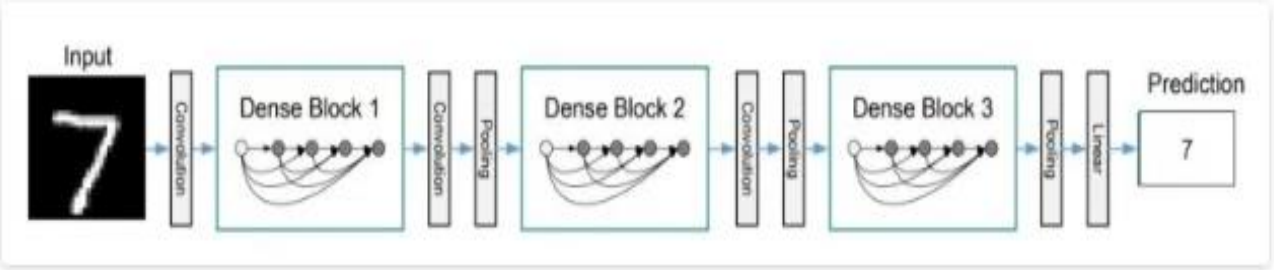
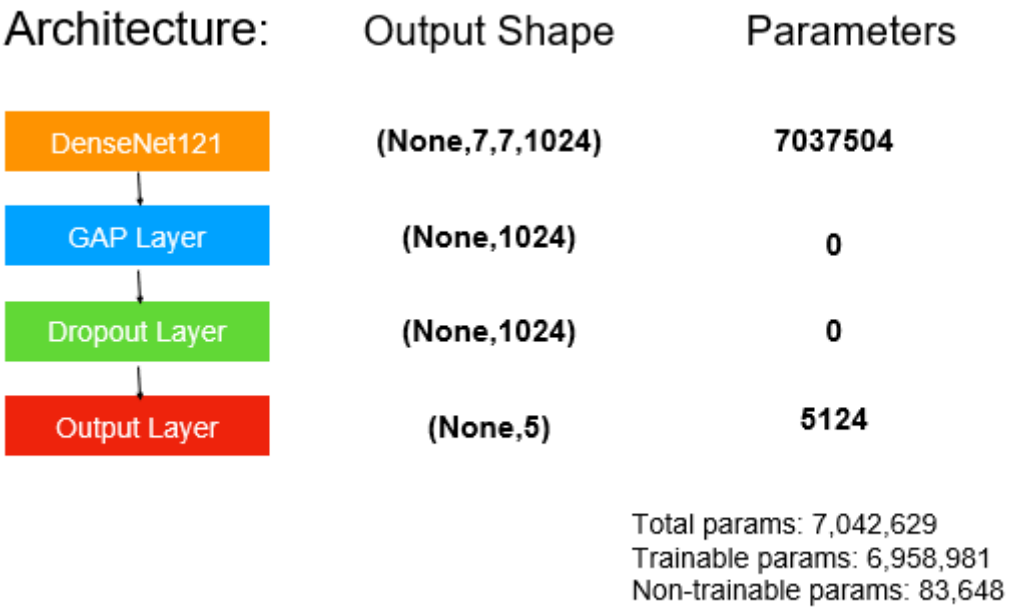


Figure 5.3: DenseNet Model Architecture

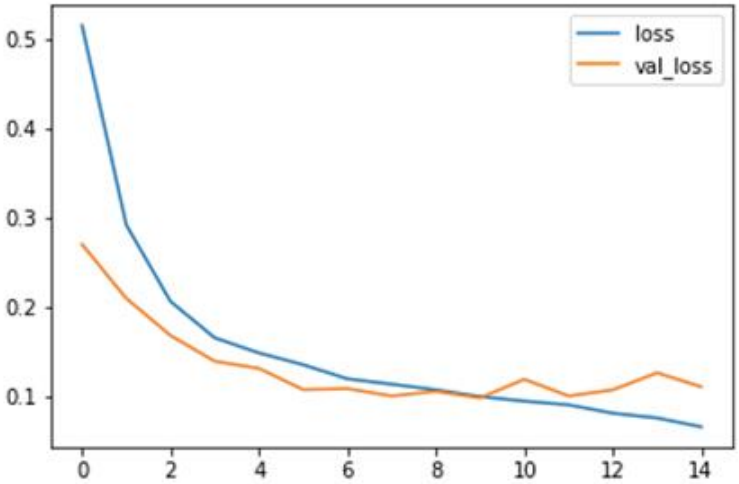
DenseNet has various variants. In this project, we have implemented 3 variants of DenseNet ie. DenseNet-121, DenseNet-169, DenseNet-201.

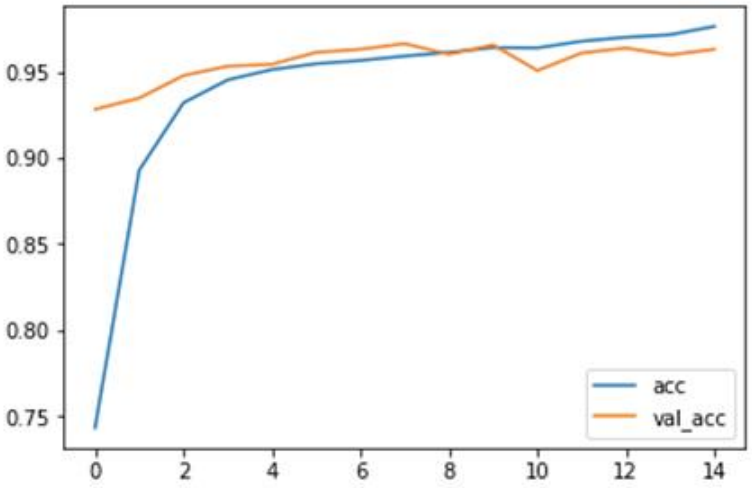
Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

DenseNet121 Model:



Training Loss & Accuracy Visualization of Densenet121





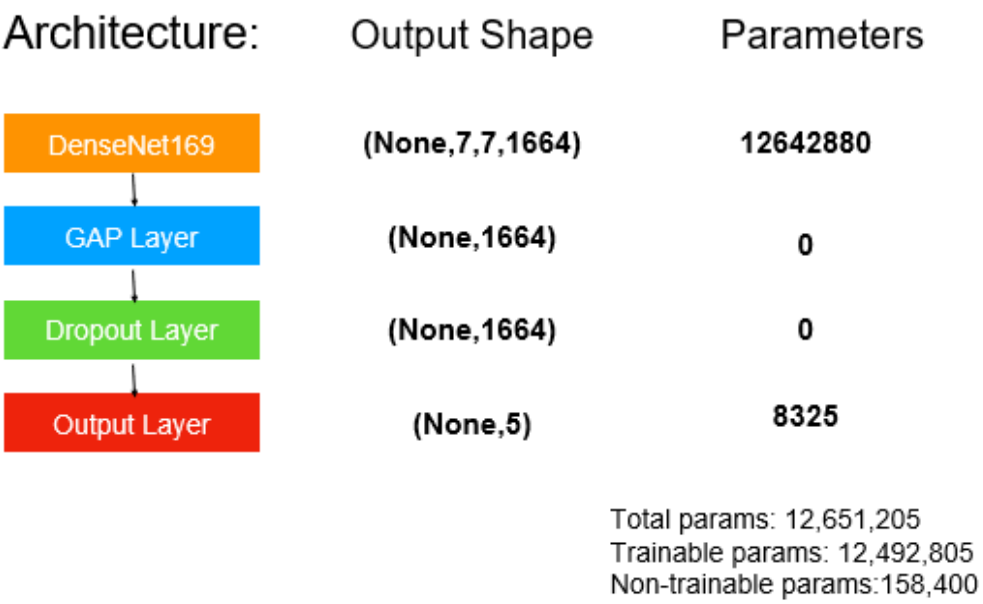
Confusion Matrix:

```
[[282   3   1   0   0]
 [  5  28  17   0   0]
 [  2   3 137   7   0]
 [  0   0  12  12   0]
 [  0   3  10  10  18]]
```

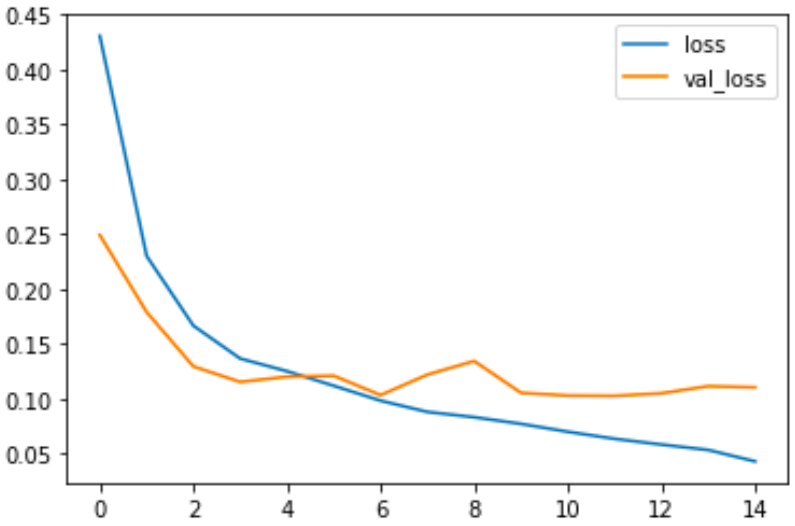
Classification Report:

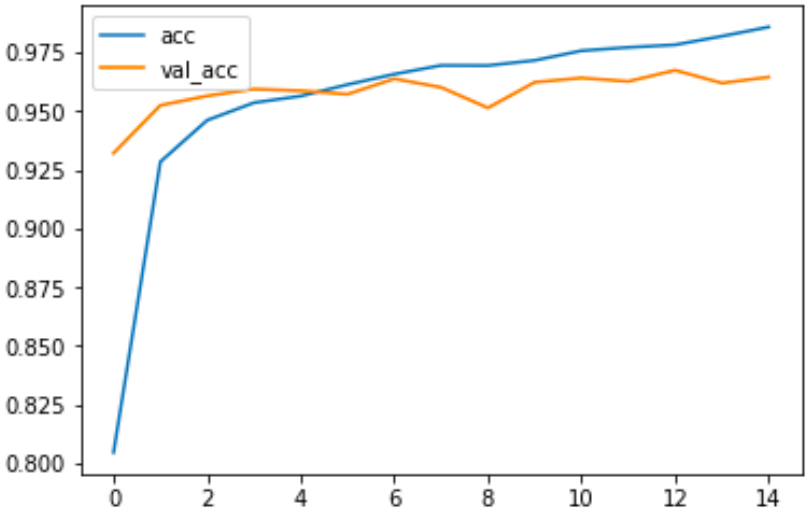
	precision	recall	f1-score	support
0	0.98	0.99	0.98	286
1	0.76	0.56	0.64	50
2	0.77	0.92	0.84	149
3	0.41	0.50	0.45	24
4	1.00	0.44	0.61	41
avg / total	0.88	0.87	0.86	550

DenseNet169 Model:



Training Loss & Accuracy Visualization of Densenet169





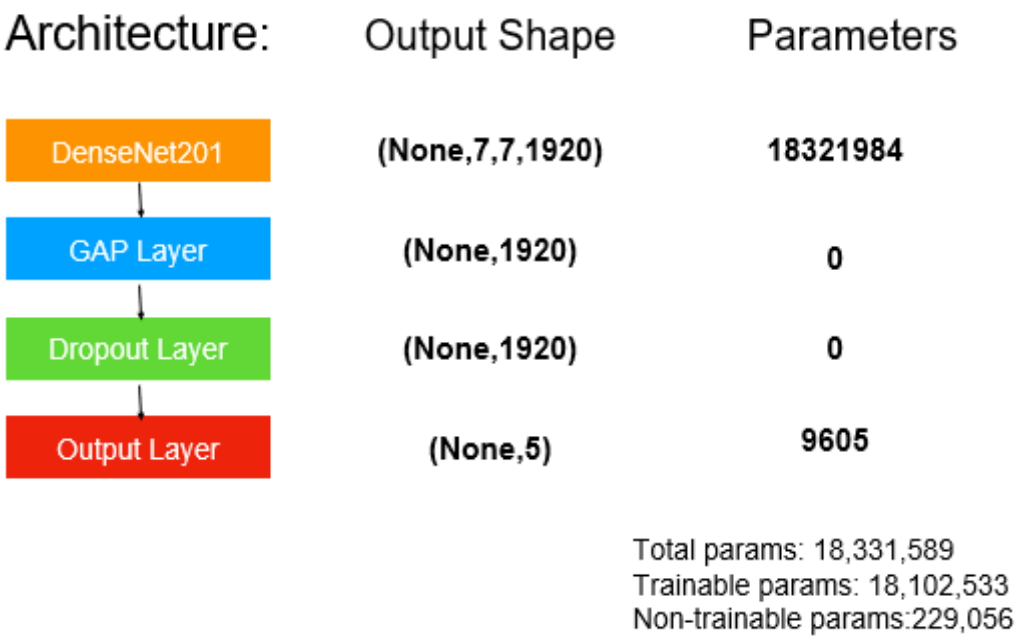
Confusion Matrix:

```
[[285   1   0   0   0]
 [  4  28  18   0   0]
 [  1   7 131  10   0]
 [  0   0   7  16   1]
 [  0   2   9  14  16]]
```

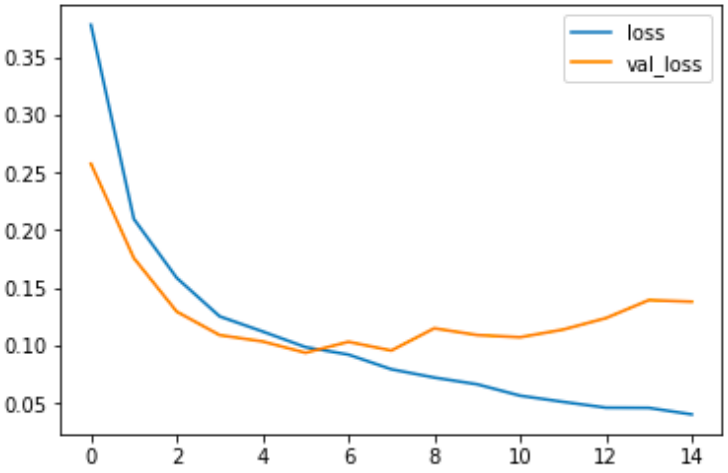
Classification Report:

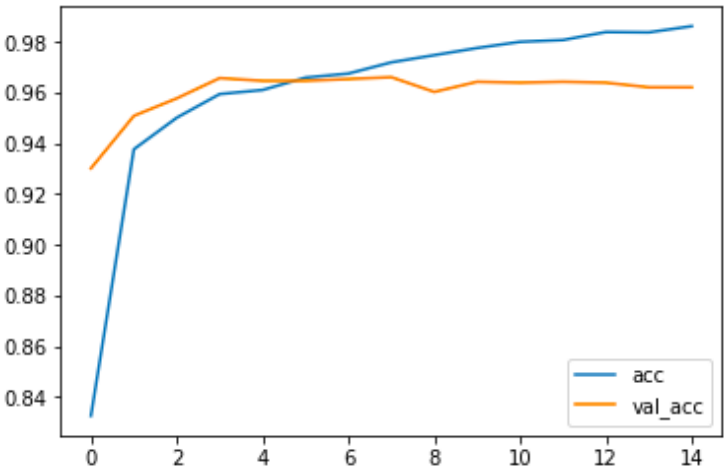
	precision	recall	f1-score	support
0	0.98	1.00	0.99	286
1	0.74	0.56	0.64	50
2	0.79	0.88	0.83	149
3	0.40	0.67	0.50	24
4	0.94	0.39	0.55	41
:al	0.88	0.87	0.86	550

DenseNet201 Model:



Training Loss & Accuracy Visualization of Densenet201





Confusion Matrix:

```
[[283   3   0   0   0]
 [  1  26  22   1   0]
 [  0   5 132  11   1]
 [  0   0  10  11   3]
 [  0   2  11  10  18]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	286
1	0.72	0.52	0.60	50
2	0.75	0.89	0.81	149
3	0.33	0.46	0.39	24
4	0.82	0.44	0.57	41
al	0.86	0.85	0.85	550

Results from Initial Experiment On The Validation Set

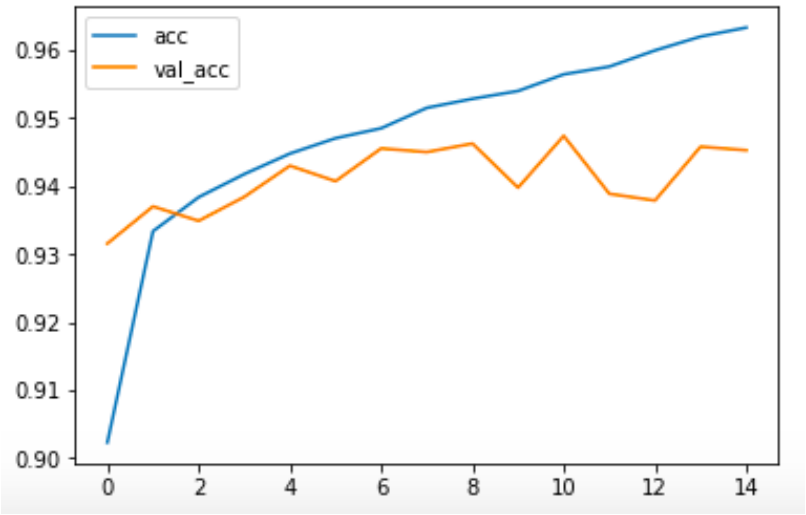
Models	DenseNet121	DenseNet169	DenseNet201
Accuracy	0.8672	0.8654	0.8545
Cohen Kappa Score	0.9172	0.9280	0.9199

All the proposed models were trained using Aptos19(3.6k) dataset to find the best model among them. From the above table, we can see that DenseNet169 gives the best result. The models were evaluated based on cohen kappa score and accuracy on the validation set.

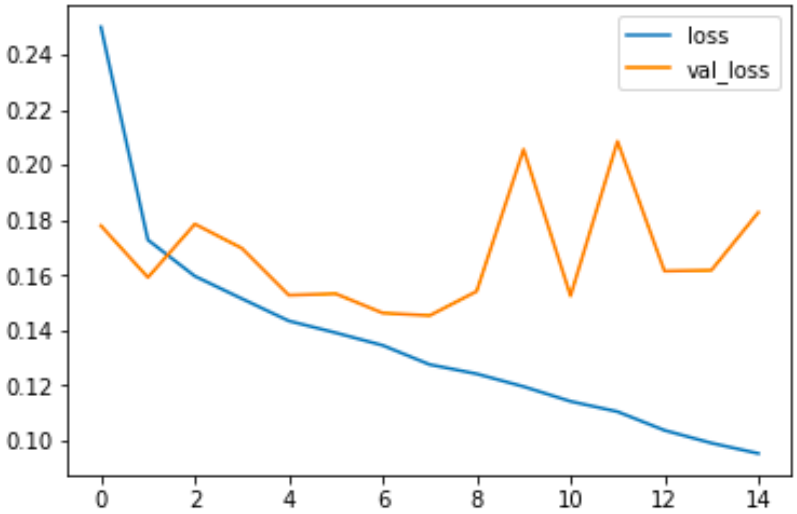
DenseNet 169 with 2015 Dataset

The best performing model was again trained with the 2015 Diabetic Retinopathy dataset.

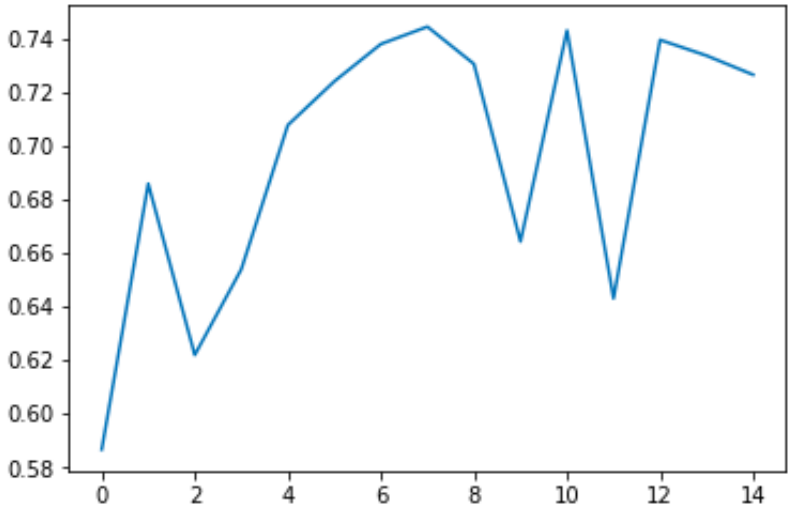
Train Vs. Validation Accuracy:



Train Vs. Validation Loss:



Cohen Kappa Score:



Confusion Matrix:

```
[[3636  123  100    3    3]
 [ 301   38   34    1    0]
 [ 226   81  453   35    3]
 [   5    2   60   43    1]
 [   7    3   23   30   58]]
```

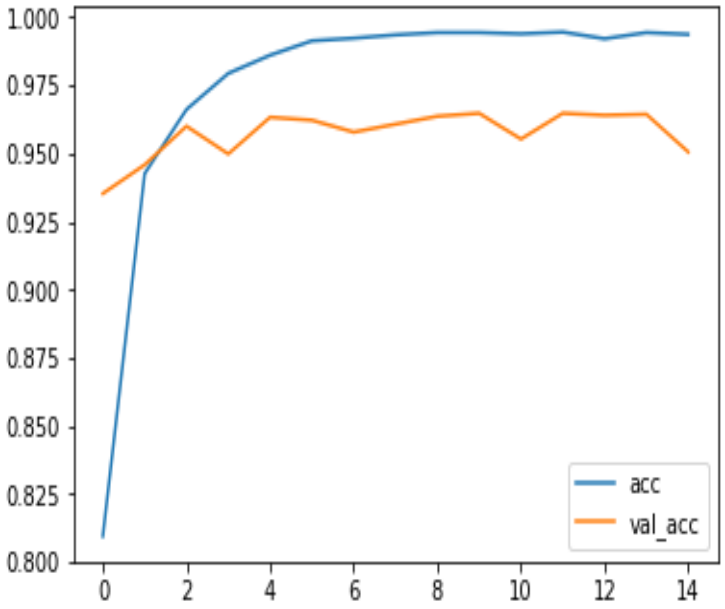
Classification Report:

	precision	recall	f1-score	support
0	0.87	0.94	0.90	39533
1	0.16	0.10	0.12	3762
2	0.64	0.55	0.59	7861
3	0.41	0.41	0.41	1214
4	0.84	0.38	0.52	1206
avg / total	0.78	0.80	0.78	53576

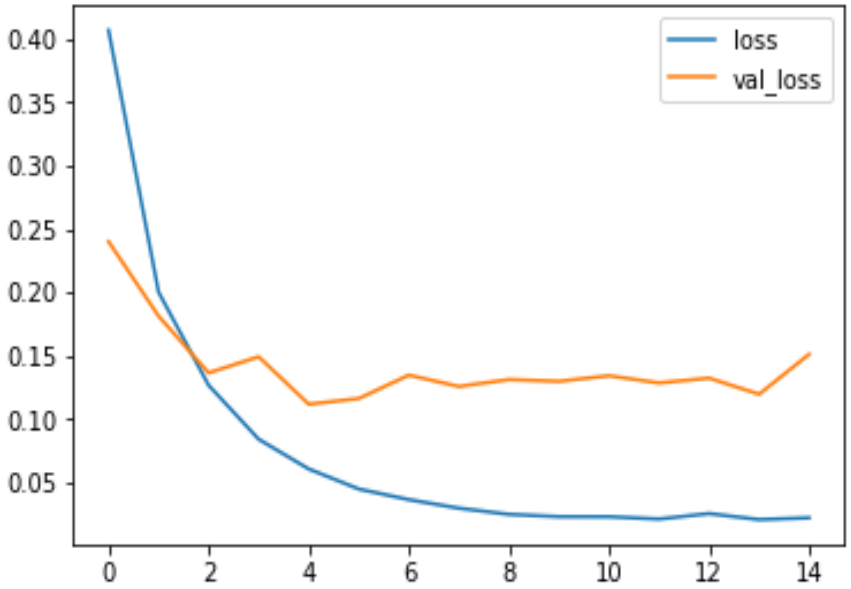
Accuracy : 0.79882783335822

DenseNet 169 without Augmentation on Aptos

Train Vs. Validation Accuracy:



Train Vs. Validation Loss:



Confusion Matrix:

```
[[280   6   0   0   0]
 [  5  30  15   0   0]
 [  0  11 130   5   3]
 [  0   0   6  14   4]
 [  0   3  11   9  18]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	286
1	0.60	0.60	0.60	50
2	0.80	0.87	0.84	149
3	0.50	0.58	0.54	24
4	0.72	0.44	0.55	41
avg / total	0.86	0.86	0.86	550

Accuracy : 0.8581818181818182

Cohen Kappa Score: 0.9144

5.6 CapsNet Model

CapsNet stands for Capsule Network. invented by Geoffrey Hinton in 2018. New architecture in neural networks and advanced approach to previous neural network design. It is a type of ann that can be used to better model hierarchical relationships. Solves the problem of vanishing gradients. It helps to preserve the position and pose information (equivariance).

Hyperparameters:

- Optimizer = Adam
- Learning Rate = 0.0001
- Batch Size = 64
- Epochs = 100

Loss Function

- Margin Loss
- Mean Squared Error (mse)

Metrics:

- Accuracy
- Out_caps

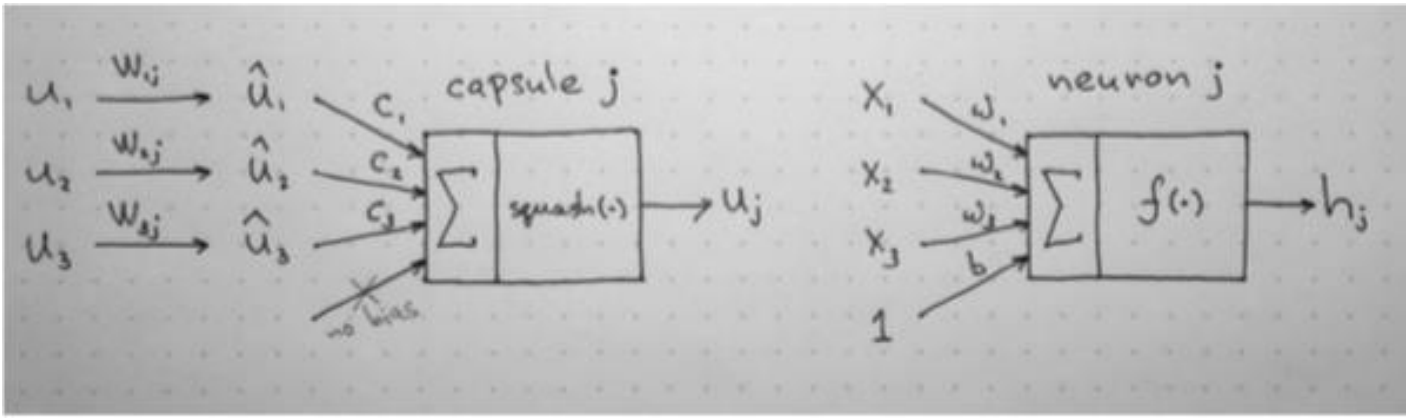


Figure5.4: CapsuleNet Model Architecture

CHAPTER 6

RESULTS

Models	**Validation Set Accuracy**	**Test Set Accuracy**
Base Model	49.13	3.46
EfficientNet Model	74.96	75.87
DenseNet Model	80.24	79.88

CHAPTER 7

SCREENSHOTS

7.1 Sample Dataset

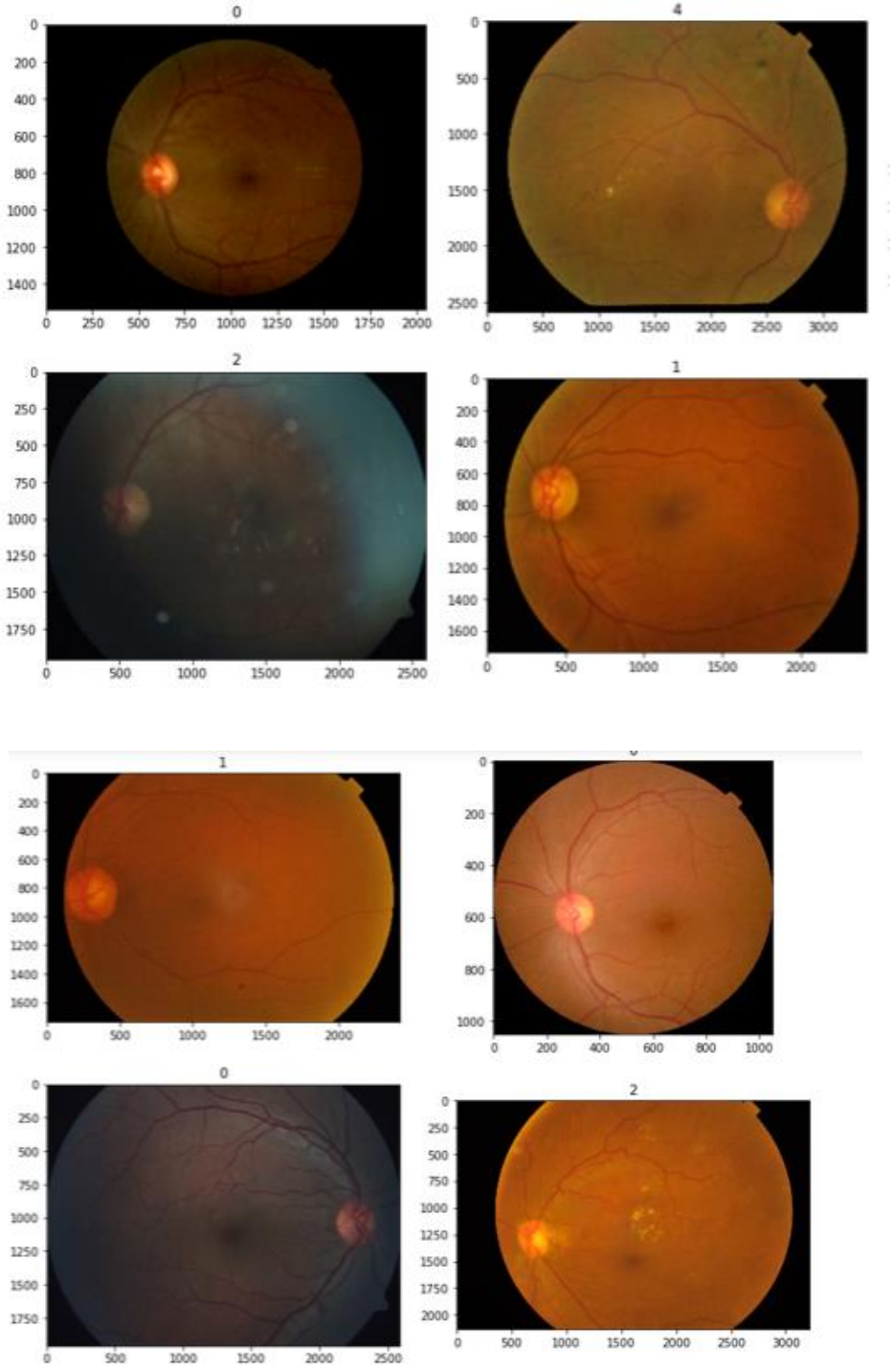


Figure 7.1: Sample Dataset

Class	Name	Number of images	Percentage
0	Normal	25810	73.48%
1	Mild NPDR	2443	6.96%
2	Moderate NPDR	5292	15.07%
3	Severe NPDR	873	2.48%
4	PDR	708	2.01%

Figure 7.2: Distribution of Dataset

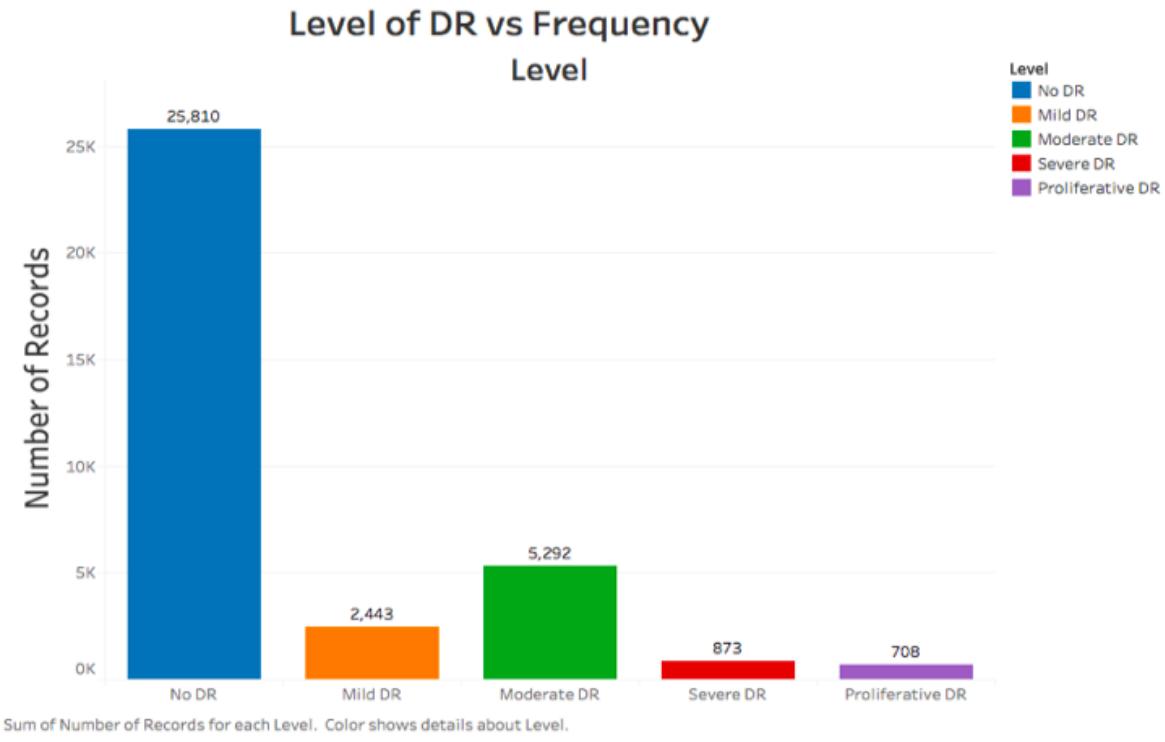


Figure 7.3: Graphical distribution of dataset

7.2 Balanced Dataset Visualization

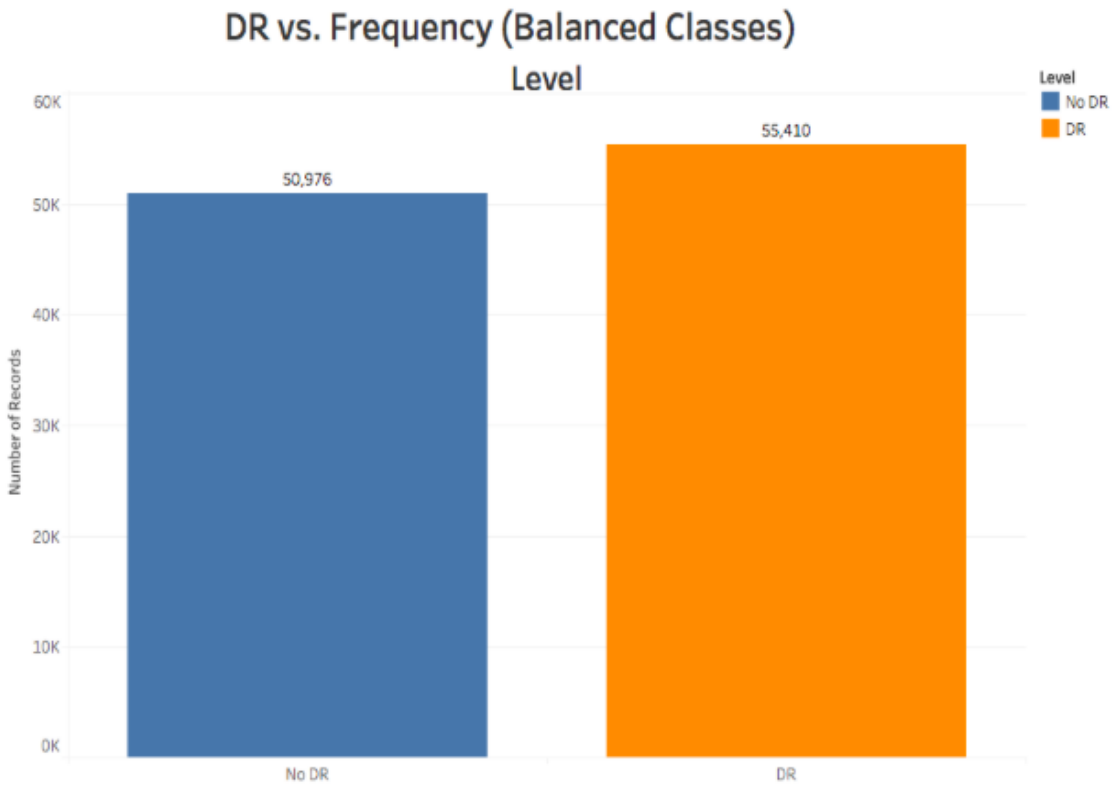


Figure 7.4: Balanced dataset graph

7.3 Retinal Images after Prepossessing and Augmentation

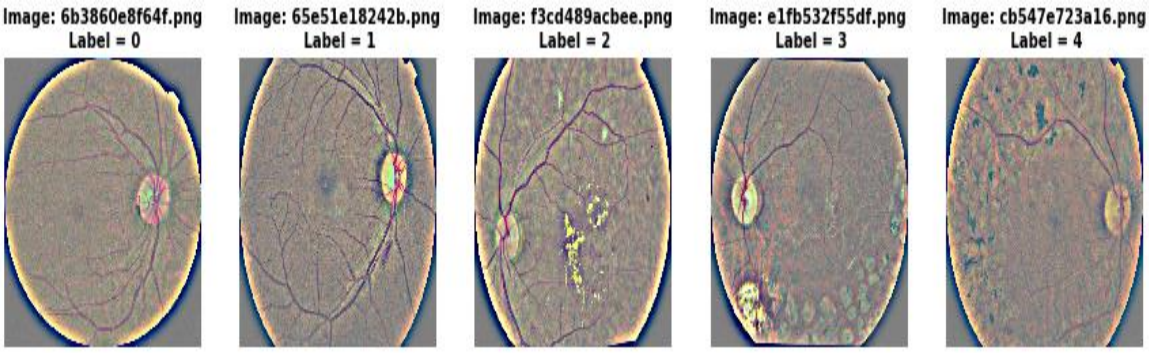


Figure 7.5: Dataset after preprocessing and augmentation

7.4 Base Model

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 94, 94, 15)	420
gaussian_dropout_2 (Gaussian	(None, 94, 94, 15)	0
conv2d_7 (Conv2D)	(None, 90, 90, 30)	11280
max_pooling2d_3 (MaxPooling2	(None, 45, 45, 30)	0
conv2d_8 (Conv2D)	(None, 43, 43, 30)	8130
max_pooling2d_4 (MaxPooling2	(None, 21, 21, 30)	0
conv2d_9 (Conv2D)	(None, 17, 17, 50)	37550
conv2d_10 (Conv2D)	(None, 11, 11, 50)	122550
dropout_2 (Dropout)	(None, 11, 11, 50)	0
flatten_2 (Flatten)	(None, 6050)	0
dense_5 (Dense)	(None, 256)	1549056
dense_6 (Dense)	(None, 128)	32896
dense_7 (Dense)	(None, 128)	16512
dense_8 (Dense)	(None, 50)	6450
dense_9 (Dense)	(None, 5)	255

Total params: 1,785,099
Trainable params: 1,785,099
Non-trainable params: 0

Figure 7.6: Base Model Summary

7.5 EfficientNet Model

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 256, 256, 3)	0	
efficientnet-b5 (Model)	(None, 8, 8, 2048)	28513520	input_3[0][0]
batch_normalization_2 (BatchNor	(None, 8, 8, 2048)	8192	efficientnet-b5[1][0]
dropout_5 (Dropout)	(None, 8, 8, 2048)	0	batch_normalization_2[0][0]
conv2d_10 (Conv2D)	(None, 8, 8, 64)	131136	dropout_5[0][0]
conv2d_11 (Conv2D)	(None, 8, 8, 16)	1040	conv2d_10[0][0]
conv2d_12 (Conv2D)	(None, 8, 8, 8)	136	conv2d_11[0][0]
conv2d_13 (Conv2D)	(None, 8, 8, 1)	9	conv2d_12[0][0]
conv2d_14 (Conv2D)	(None, 8, 8, 2048)	2048	conv2d_13[0][0]
multiply_2 (Multiply)	(None, 8, 8, 2048)	0	conv2d_14[0][0] batch_normalization_2[0][0]
global_average_pooling2d_3 (Glo	(None, 2048)	0	multiply_2[0][0]
global_average_pooling2d_4 (Glo	(None, 2048)	0	conv2d_14[0][0]
RescaleGAP (Lambda)	(None, 2048)	0	global_average_pooling2d_3[0][0] global_average_pooling2d_4[0][0]
dropout_6 (Dropout)	(None, 2048)	0	RescaleGAP[0][0]
dense_3 (Dense)	(None, 128)	262272	dropout_6[0][0]
dropout_7 (Dropout)	(None, 128)	0	dense_3[0][0]
dense_4 (Dense)	(None, 5)	645	dropout_7[0][0]

Figure 7.7: EfficientNet Model Summary

7.6 DenseNet Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Model)          (None, 10, 10, 1024)      7037504
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1024)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 5125
=================================================================
Total params: 7,042,629
Trainable params: 6,958,981
Non-trainable params: 83,648
_________________________________________________________________
```

Figure 7.8: DenseNet Model Summary

7.7 DenseNet model output graph

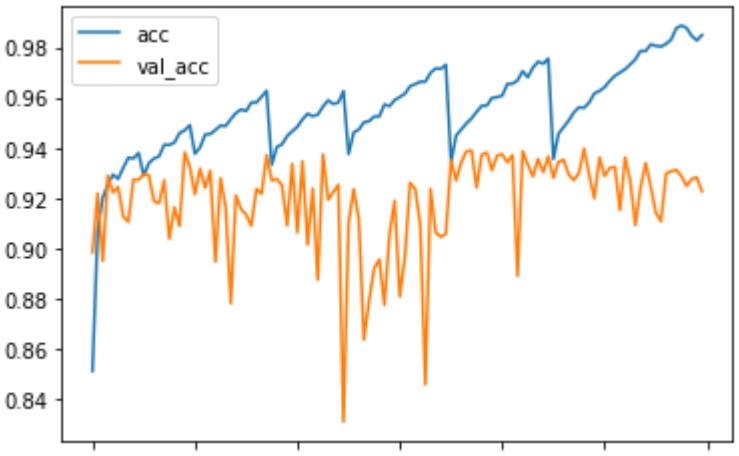


Figure 7.9: DenseNet Accuracy v/s Validation Accuracy graph

7.8 CapsNet Model

```
__________________________________________________________________________
__________________________
Layer (type)                 Output Shape              Param #     Connect
ed to
==========================================================================
==========================
input_1 (InputLayer)         (None, 32, 32, 3)         0
__________________________________________________________________________
__________________________
conv1 (Conv2D)               (None, 24, 24, 256)       62464       input_1
[0][0]
__________________________________________________________________________
__________________________
conv2d_1 (Conv2D)            (None, 8, 8, 256)         5308672     conv1
[0][0]
__________________________________________________________________________
__________________________
reshape_1 (Reshape)          (None, 2048, 8)           0           conv2d_
1[0][0]
__________________________________________________________________________
__________________________
```

```
lambda_1 (Lambda)            (None, 2048, 8)           0           reshape
_1[0][0]
__________________________________________________________________________
__________________________
digitcaps (CapsuleLayer)     (None, 5, 16)             1320960     lambda_
1[0][0]
__________________________________________________________________________
__________________________
input_2 (InputLayer)         (None, 5)                 0
__________________________________________________________________________
__________________________
mask_1 (Mask)                (None, 16)                0           digitca
ps[0][0]
                                                                   input_2
[0][0]
__________________________________________________________________________
__________________________
dense_1 (Dense)              (None, 512)               8704        mask_1
[0][0]
```

```
__________________________
dense_2 (Dense)                 (None, 1024)         525312      dense_1
[0][0]
__________________________________________________________________________
__________________________
dense_3 (Dense)                 (None, 3072)         3148800     dense_2
[0][0]
__________________________________________________________________________
__________________________
out_caps (Length)               (None, 5)            0           digitca
ps[0][0]
__________________________________________________________________________
__________________________
out_recon (Reshape)             (None, 32, 32, 3)    0           dense_3
[0][0]
==========================================================================
==========================
Total params: 10,374,912
Trainable params: 10,364,672
Non-trainable params: 10,240
__________________________________________________________________________
```

Figure 7.10: CapsuleNet Summary

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

Diabetic Retinopathy(DR) is one of the major causes of blindness in the present world. If detected early enough, effective treatment of DR is available, making this a vital process. In this project, we have implemented a deep learning model that takes retinal images as input and detects the presence of DR, and also classifies them into 5 different levels of DR.We have implemented 4 CNN architectures namely base model, EfficientNet model, DenseNet model, and CapsNet model. We achieved a good amount of accuracy with EfficientNet and DenseNet models. The EfficientNet model got the highest accuracy of 88%.

8.2 FUTURE WORK

The system that we have currently developed has multiple CNN architecture that can detect the presence of DR and classify them into different levels of DR. Further, we want to focus more on making the dataset more balanced, by using different data preprocessing and data augmentation methods. We plan to focus on improving the overall performance of the system. We plan to improve the accuracy of the models by using different hyperparameters.

REFERENCES

[1] https://www.medicalnewstoday.com/articles/183417

[2] https://www.kaggle.com/search?q=diabetic+retinopathy+in%3Adatasets

[3] https://www.w3schools.com/about/

[4] https://blog.coursera.org/about/

[5] https://www.learninglight.com/best-elearning-white-papers/

[6]https:// www.ncbi.nlm.nih.gov/pmc/articles/PMC3874488/_

[7] http://www.uptodate.com/contents/diabetic-retinopathy-classification-and-clinical-features

[8] https://www.nature.com/articles/s41746-019-0172-3

[9] https:// www.sciencedirect.com/science/article/pii/S1877050916311929

[10] https://www.tutorialspoint.com/python/python_overview.htm

[11] https://www.modernretina.com/dme/deep-learning-future-diabetic-retinopathy-screening

[12]https:// towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53-

[13]https://www.pythonforengineers.com

[14] http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[15] https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[16]https:// www.coursera.org/learn/convolutional-neural-networks

[17] https://www.researchgate.net/publication/257948587_

[18]http://shodhganga.inflibnet.ac.in/bitstream/10603/34521/6/06_chapter1.pdf

[19] https://www.economicsnetwork.ac.uk/cheer/ch18/manochehr.pdf

[20] https://towardsdatascience.com/data-augmentation-for-deep-learning-4fe21d1a4eb9

[21] https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825

[22] https://towardsdatascience.com/neural-network-architectures-156e5bad51ba

[23] https://www.jeremyjordan.me/convnet-architectures/

[24] https://medium.com/@lessw/efficientnet-from-google-optimally-scaling-cnn-model-architectures-with-compound-scaling-e094d84d19d4

[25] http://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

[26] https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a

[27] https://medium.com/@smallfishbigsea/densenet-2b0889854a92

[28] https://towardsdatascience.com/capsule-networks-the-new-deep-learning-network-bd917e6818e8

[29] https://medium.com/ai%C2%B3-theory-practice-business/part-iv-capsnet-architecture-6a64422f7dce