

Diabetic Retinopathy Detection and Classification by Using Deep Learning

by

Shahriar Hossain

21141036

Md. Nurussafi Evan

18101525

Fariya Zakir Farhin

18101505

Mashrur Karim Nabil

19101659

Sameen Sadman

21341058

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2022

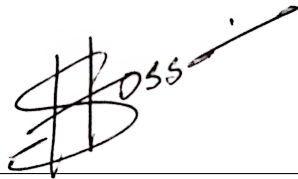
© 2022. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



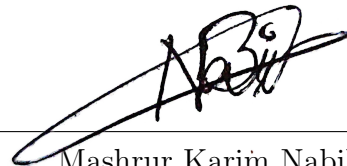
Shahriar Hossain
21141036



Md. Nurushafi Evan
18101525



Fariya Zakir Farhin
18101505



Mashrur Karim Nabil
19101659



Sameen Sadman
21341058

Approval

The thesis titled “Diabetic Retinopathy Detection and Classification by Using Deep Learning” submitted by

1. Shahriar Hossain (21141036)
2. Md. Nurushafi Evan (18101525)
3. Fariya Zakir Farhin (18101505)
4. Mashrur Karim Nabil (19101659)
5. Sameen Sadman (21341058)

Of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 18, 2022.

Examining Committee:

Supervisor:
(Member)



Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

Hereby, we, the members, consciously assure that the following is fulfilled for the manuscript, "Diabetic Retinopathy Detection and Classification by Using Deep Learning".

1. The contents of this paper are unique work of the writers' and it has not been published before.
2. Only the authors' own research and analysis is presented in the work with utmost accuracy.
3. Contributions by the co-authors and co-researchers are rightly acknowledged in the study.
4. All sources are appropriately mentioned (correct citation). Text that is literally copied must be identified as such by using quote marks and providing suitable reference.
5. The authors actively participated and put in effort leading to the article and any public responsibility related to it's content will be accepted.

Violations of the Ethical Statement standards may have serious repercussions. We agree with the preceding declarations and certify that this submission adheres to BRAC University's rules.

Abstract

Eyes are the most sensitive part of a human being and it is one of the most challenging tasks for a computer-aided system to classify its diseases. Many vision-threatening diseases such as, Glaucoma and Diabetic Retinopathy are treated using digital fundus imaging and retinal images by the specialist at a primary level. However, a computer-aided system that can classify if the eye has a disease or not could be a handy tool for the specialists and a challenging task for computer aided system developers. A branch of machine learning which is deep learning is making a revolutionary impact on medical diagnosis using image processing and pattern recognition. Therefore, we aim to make use of some Convolutional Neural Network (CNN) architectures such as ResNet50, Inception V3, Xception, DenseNet-169 and MobileNetV3 Large to extract the features and classify if the eye has a disease or not using digital fundus photography and retinal image. For our research, we used a competition dataset available from Kaggle [1] and another dataset from IDRiD [2]. Our final dataset contained a total of 2,517 images with each stage having around 500 images in them. Upon training and testing the selected architectures, we have found that Inception V3 has an accuracy of 86.31% and 87.7% (with a lowered learning rate). Similarly for Xception, we attained 86.9% accuracy with default learning rate and 87.9% accuracy with lowered learning rate. ResNet50 gave an accuracy of 46.83%, MobileNetV3 Large gave the lowest accuracy standing at 23.81%. DenseNet-169 gave us the highest accuracy among all other models, soaring at 88.29% accuracy.

Keywords: Convolutional Neural Network, Deep Learning, Diabetic Retinopathy, InceptionV3, Xception, DenseNet-169

Dedication

This research is dedicated to our mentors. Our beloved mentors who guided us the entire time and without their help, we would not have been able to complete this thesis. We received crucial guidance and help alongside all the academic knowledge they have imparted upon us.

Acknowledgement

Firstly, all praise to the Almighty Allah for whom our thesis have been completed on time and without any major interruption.

We would like to express our deepest and sincerest gratitude to Dr. Amitabha Chakrabarty, our esteemed instructor and advisor for his guidance throughout our work.

And finally to our dearest parents. Without their relentless support, we might not have been able to complete this paper. Thanks to their kindness and efforts, we are at the verge of our graduation.

Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iv
Abstract	v
Dedication	vi
Acknowledgment	vii
Table of Contents	viii
List of Figures	x
Nomenclature	xii
1 Introduction	1
1.1 Motivation	1
1.2 History of Diabetic Retinopathy	1
1.3 Research Objective	2
1.4 Research Methodology	3
1.5 Research Orientation	3
2 Related Work	4
3 Research Methodology	6
3.1 Data Pre-Processing	7
3.1.1 Data Cleaning	8
3.1.2 Data Handling	8
3.1.3 Making Hybrid Dataset	10
3.1.4 Image Enhancement using Contrast Limited Adaptive His- togram Equalization (CLAHE)	11
3.1.5 Augmentation	12
3.2 Convolutional Neural Networks	13
3.3 Model Training	14
3.3.1 ResNet50	15
3.3.2 Inception or GoogLeNet	15
3.3.3 MobileNetV3 Large	16

3.3.4	Xception	17
3.3.5	DenseNet-169	19
3.4	Adamax Optimizer	20
3.5	Hyper-Parameter Tuning	20
4	Implementation	22
4.1	Dataset	22
4.1.1	Source	22
4.1.2	Dataset Description	22
4.1.3	Data Sample	23
4.2	Applying Deep Learning Algorithms	26
4.2.1	Train Test Split	27
4.2.2	Taking Image Input	28
4.2.3	Augmentation	28
4.2.4	Our CNN Model	29
4.2.5	Compiling the model	30
4.2.6	Optimizer and Hyper-Parameter Tuning	31
4.2.7	Fitting the model	31
4.2.8	Model Check Point Call Back Function	31
4.2.9	Model Evaluation	31
5	Results and Analysis	34
5.1	Result Overview	34
5.2	Model Fit	34
5.2.1	Underfit Model	34
5.2.2	Overfit Model	35
5.2.3	Optimum Model	36
5.3	Validation Loss & Accuracy Comparison With Different Learning Rate	36
5.3.1	Accuracy Comparison	37
5.3.2	Loss Comparison	38
5.4	InceptionV3 With Default Learning Rate	39
5.5	InceptionV3 With Lowered Learning Rate	41
5.6	Xception With Default Learning Rate	42
5.7	Xception With Lowered Learning Rate	44
5.8	ResNet With Lowered Learning Rate	45
5.9	DenseNet-169 With Lowered Learning Rate	47
5.10	MobileNetV3 Large With Lowered Learning Rate	49
5.11	Comparison & Verdict	50
6	Conclusion	52
	Bibliography	55

List of Figures

3.1	Workflow Diagram	7
3.2	Truncated Image and Bad Image	8
3.3	Imbalanced Data (APTOS)	9
3.4	Imbalanced Data (IDRID)	9
3.5	Bad Images	11
3.6	Hybrid Dataset Count Visualized	11
3.7	Retinal Images before and after using CLAHE	12
3.8	Convolutional Neural Network	14
3.9	Residual Learning: Building Block	15
3.10	Layers of InceptionV3	16
3.11	Simplified InceptionV3	16
3.12	MobileNetV2: Bottleneck with residual	17
3.13	MobileNetV3 Block	17
3.14	MobileNetV3 Large Structure and MobileNetV3 Small Structure	17
3.15	Xception Workflow	18
3.16	Xception vs Other Models in Imagenet Dataset	18
3.17	Xception vs Inception V3	19
3.18	5-layer dense block with a growth rate of $k = 4$ and the standard ResNet structure	20
3.19	Adamax Formula	20
4.1	Sample Data	23
4.2	Level - 0: No DR	25
4.3	Level - 1: Mild DR	25
4.4	Level - 2: Moderate DR	25
4.5	Level - 3: Severe DR	26
4.6	Level - 4: Proliferative DR	26
4.7	Output of ImageDataGenerator	28
4.8	DenseNet169 Architecture	30
4.9	DenseNet169 Architecture Loss Accuracy Curve	32
4.10	DenseNet169 model's Confusion Matrix	33
4.11	DenseNet169 model's Classification Report	33
5.1	Underfit	35
5.2	Overfit	35
5.3	Optimum	36
5.4	Validation Loss & Accuracy Comparison	37
5.5	Accuracy Comparison	38
5.6	Validation Loss	39

5.7	InceptionV3 with Learning Rate 0.001	39
5.8	InceptionV3 Confusion Matrix Default Learning Rate	40
5.9	InceptionV3 Loss-Accuracy Curve (Default)	40
5.10	InceptionV3 with Learning Rate 0.0001	41
5.11	InceptionV3 Confusion Matrix Lowered Learning Rate	41
5.12	InceptionV3 Loss-Accuracy Curve (Default)	42
5.13	Xception with Learning Rate 0.001	42
5.14	InceptionV3 Confusion Matrix Default Learning Rate	43
5.15	Xception Loss-Accuracy Curve (Default)	43
5.16	Xception with Learning Rate 0.0001	44
5.17	InceptionV3 Confusion Matrix Lowered Learning Rate	44
5.18	Xception Loss-Accuracy Curve (Lowered)	45
5.19	ResNet with Learning Rate 0.0001	46
5.20	ResNet Confusion Matrix Lowered Learning Rate	46
5.21	ResNet Loss-Accuracy Curve (Lowered)	47
5.22	DenseNet-169 with Learning Rate 0.0001	47
5.23	DenseNet-169 Confusion Matrix Lowered Learning Rate	48
5.24	DenseNet-169 Loss-Accuracy Curve (Lowered)	48
5.25	MobileNetV3 Large with Learning Rate 0.0001	49
5.26	MobileNetV3 Large Confusion Matrix Lowered Learning Rate	49
5.27	MobileNetV3 Large Loss-Accuracy Curve (Lowered)	50

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CLAHE Contrast Limited Adaptive Histogram Equalization

CNN Convolutional Neural Network

CNN Convolutional Neural Network

CSV Comma Separated Values

DNN Deep Neural Network

DR Diabetic Retinopathy

GMM Gaussian Mixture Model

IDRiD Indian Diabetic Retinopathy Image Dataset

ILSVRC ImageNet Large Scale Visual Recognition Challenge

MA Microaneurysms

OCT Optical Coherence Tomography

PCA Principal Component Analysis

px Pixel

RBM Restricted Boltzmann Machine

RGB Red Green Blue

SIFT Spatial invariant feature transform

SVD Singular Value Decomposition

SVM Support Vector Machines

VGG – 16 Visual Geometry Group - 16

Chapter 1

Introduction

1.1 Motivation

One of the most vital organs in the human body is the eye. All living organisms have eyes that allow them to view the world around them. This complicated physical part allows us to perceive things, which we call vision. Light rays enter the eyes through the cornea after reflecting from various things, and are then processed by the brain to generate an image of that item. One of our most crucial senses is sight, which accounts for 80 percent of what we see. Our way of living has a big influence on how well our eyes are doing. Everything adds to the rise in the risk of eye problems: stress, eating habits, sleep, diabetes, and so on. Visual impairment is a national and international health issue that has a severe influence on both physical and mental health. Visually handicapped people are more likely to develop chronic illnesses, have accidents, withdraw from social situations, get depressed, and even might face untimely death. Because of the aging population, the number of persons with vision impairment and blindness is growing. Near- or far-sightedness affects at least 2.2 billion individuals worldwide. Vision impairment might have been averted or handled in at least 1 billion of these instances, or about half of them. Not taking proper care of our eyes can lead to permanent blindness. Medical science and technology have advanced by leaps and bounds over the years. This opened doors to new ways of imaging and screening of eye diseases. Usage of AI and automated predictions in combination with medical knowledge is gaining traction rapidly [3]–[5]. Medical institutes and organizations around the globe collect, analyzes and performs research on this data to fine-tune medical procedures. Storing medical information in electronic form has great potential as well. Such data can be transferred to different medical experts almost instantly to ensure greater healthcare [6]. These data can be processed through many prominent machine learning models to automate and increase the accuracy of disease detection based on training set or data. If the same was to be analyzed manually, it would take ages and the human margin of error is too high to depend on. We believe our deep learning model is a solid solution which will help in making more accurate and swift decision in such eye disease cases.

1.2 History of Diabetic Retinopathy

Eye disorders are prevalent, and the reasons of these diseases can be caused by a variety of variables such as external or internal injury, advanced age, or a variety

of maladies. Many eye illnesses, such as glaucoma, diabetic retinopathy, diabetic macular edema, and melanoma, can result in permanent vision loss.

Early detection and treatment of such disorders can spare one's eyes from these diseases. Any ophthalmologist would benefit from a computer-aided system that can diagnose these disorders in real time. It might help prevent one's vision from becoming blurred.

DR lesions however are not easy to detect and classify. In [7], the authors proposed a new hybrid classifier. In the paper, an m-Mediods and GMM (Gaussian Mixture Model) classifier was used as an ensemble. This ensured that the best properties from each classifier comes into account when the architectures process on the dataset.

Clinical data has previously been used to point out relational signs in order to diagnose eye diseases [8]. Our goal is to make such a system which will help the medical sector especially the ophthalmologist with a time efficient multi disease prediction with the help of the Deep learning.

Color fundus photography was utilized to detect diabetic retinopathy in the study [9]. For their investigation, the authors employed the InceptionV3 architecture. From the dataset provided, InceptionV3 extracts both general (5x5) and local (1x1) features at the same time. They have had remarkable success using the OpenCV library for picture pre-processing, loading, and manipulations.

We've seen cases where the large color range of fundus pictures was addressed by normalizing such retinal images using various histogram specification approaches [10]. This approach was used to standardize all of the photos within the same parameters, making pre-processing and model training more exact. Computer-based medical tests are frequently chastised, and the issue is exacerbated when it comes to categorizing and identifying pictures based on patterns as well as colors.

1.3 Research Objective

We want to leverage widely available medical data and machine learning methods to create a model that can properly predict diabetic retinopathy with high accuracy. Hospitals have access to all patient data required for the study, including age, gender, weight, diabetes level, fasting blood sugar, and an image of the patient's eyes. Most medical clinics now use complex frameworks to store this critical data in their databases. Eye illnesses are diagnosed clinically by professionals with competence and experience. However, patients must undergo a series of costly testing.

We used machine learning methods to identify Diabetic Retinopathy and its various phases in this study. We'll assess the accuracy of each algorithm's output independently to have a better understanding of which algorithms are best for fundoscopic image data.

To identify DR from fundoscopy pictures, convolutional neural networks were used.

CNNs have been employed by several academics to perform similar jobs [11]. Some of these models have also shown to be quite accurate.

We have discovered a low-cost method of accurately diagnosing diabetic retinopathy after additional research and refinement of our model. Deep learning architectures trained on funduscopy pictures have shown to be fairly accurate in recognizing the disorders in question. On funduscopy pictures, we'll test a variety of deep learning architectures. For Diabetic Retinopathy, all of the selected architectures will be run separately. We will select the best performing architecture after model training to meet our aim of properly recognizing these illnesses at an early stage.

1.4 Research Methodology

This model is responsible to accurately predict if a patient has diabetic retinopathy or not depending on the patients fundoscopic images. The proposed model makes use of various data pre-processing techniques such as Image Augmentation, CLAHE implementation, Hybridization and etc. Finally, utilizing various deep learning architectures such as DenseNet-169 and etc. we get a prediction on the test data. The prediction comes with both if the patient has diabetic retinopathy or not and at what stage it is currently.

1.5 Research Orientation

We already mentioned study and effort on Diabetic Retinopathy by other researchers in section 2. Following that, part 3 delves into the deep learning architectures, methodology, and procedures that we employed to conduct our own study on the subject. In section 4, we illustrate how we used our research strategy and methodology to develop the previously discussed algorithms and structures. There are other examples of our dataset and pre-processing procedures in this section. The findings of our research, as well as the results and analysis of the data obtained after implementing our recommended techniques, are presented in section 5. Finally, section 6 contains the recommended research's future work goals and a thorough conclusion.

Chapter 2

Related Work

Andrés Ortiz, Jorge Munilla, Juan M Górriz, and Javier Ramirez created a groundbreaking approach for detecting Alzheimer's disease early on [12]. They used automated anatomical labeling (AAL) to train a deep belief network to designate various areas of MRI (Magnetic Resonance Imaging) and PET (Positron Emission Tomography). To assess data intensity, these voxelized pictures with specified area markers are processed through an unsupervised network using Restricted Boltzmann Machines with a two-layer architecture. For regression and classification, trained RBM and SVM [13] are utilized to correctly detect early stages of Alzheimer's disease. The main idea behind voxelizing pictures is to mark out distinct regions and then use our trained model to detect eye illness from retinal images.

The authors present a computer-based automatic classification algorithm to identify Age-related macular degeneration AMD. 1,20,656 manually graded color fundus images from AREDS study was collected for this study [14]. The authors defined a severity scale of 13 class to classify AMD. After pre-processing the images, multiple convolution neural nets (CNNs) were trained independently. CNN was trained to optimize evaluation metric by comparing the CNN output with the true class iteratively and then adjusting the weights to minimize the loss between CNN output and actual label. A random forest algorithm was trained to build a model ensemble based on the results of the single CNNs. Performance of the algorithm was evaluated on 5555 independent fundus images from the KORA study. The deep learning algorithm outperformed human graders in the AREDS study and is suitable to classify AMD fundus images in other datasets.

In this paper, the authors tried to create a system that will act as a referral trigger, which would advise the patient to consult a retinal expert upon positive detection [15]. This study included two phases in which the model was trained and tested in the first phase and a GUI was developed for real-time detection in the second phase. They divided the dataset into an 8:2 ratio for training and testing. Features were extracted from the fundus images through a series of convolution, pooling, ReLu layers. An 80% accuracy was acquired which can be further improved by performing parameter tuning and adopting cross-validation.

The author implemented GPU accelerated Deep CNN to automatically detect Diabetic Retinopathy from fundus color images. They classified the dataset images into

5 stages of severity. The authors three major CNN models and obtained a max accuracy of 38.66% [16]. The test set contained 5,000 fundus images. The examination in diagnosing diabetic retinopathy has been founded on the extraction of highlights from the images like microaneurysms and injuries through which the classification is performed. This work was implemented using ImageMagick and python library, OpenCV.

A. Alaimahal, Dr. S. Vasuki developed a system to help ophthalmologists in the screening process of Diabetic Retinopathy. The system concentrates on detecting microaneurysm from fundus images [16]. The fundus images were pre-processed to reduce noise and increase contrast for easier detection of microaneurysms. The sensitivity and Predictive value of the proposed system is 98.89% and 89.70% respectively. It was also able to detect microaneurysms from poor quality images.

Wejdan L. Alyoubi, Wafaa M. Shalash, Maysoon F. Abulkhair together has reviewed 33 papers in their given paper named Diabetic retinopathy detection through deep learning techniques: A review. The authors describe the 5 stages of diabetic retinopathy which are No DR, mild DR, moderate DR, severe DR and proliferative DR. They also found that many CNN methods can be used to detect Diabetic Retinopathy. Such as, estricted Boltzmann Machines, CNN, auto encoder and sparse coding [17]. Performance is proportional to the quantity of training data. The Authors also illustrated that, SoftMax activation function is the most used classification function. The authors also included that, Fundus color images and optical coherence tomography (OCT) are used as the datasets. Lastly They added that, Many CNN can be used for the detection eg. AlexNet, VGG-16, SqueezeNet, ResNet50, Inception V3, InceptionResNetV2, Xception, DenseNets etc.

Daniel Shu Wei Ting, Carol Yim-Lui Cheung, Gilbert Lim, et al. in their paper has proposed a model to evaluate diabetic retinopathy with great accuracy. This model extracts data from retinal images to predict the severity of diabetic retinopathy with high accuracy [18]. The deep learning system shows an accuracy of upto 93.2%.

Tahira Nazir, Aun Irtaza, Ali Javed, Hafiz Malik, Dildar Hussain, Rizwan Ali Naqvi has developed deep learning system to analyze and detect diabetic retinopathy. Fast Region based Convulational Neural Network (FRCNN) and Fuzzy K-Means (FKM) was used to process images and localize different regions for the classes and states of the disease [19]. The model has shown an accuracy of 95.8%.

Chapter 3

Research Methodology

Our Deep Learning model has been fine-tuned to identify Diabetic Retinopathy at various stages. Various data pre-processing techniques and deep learning architectures are employed in our model.

The following flow chart (fig 3.1) summarizes the full work process of our investigation.

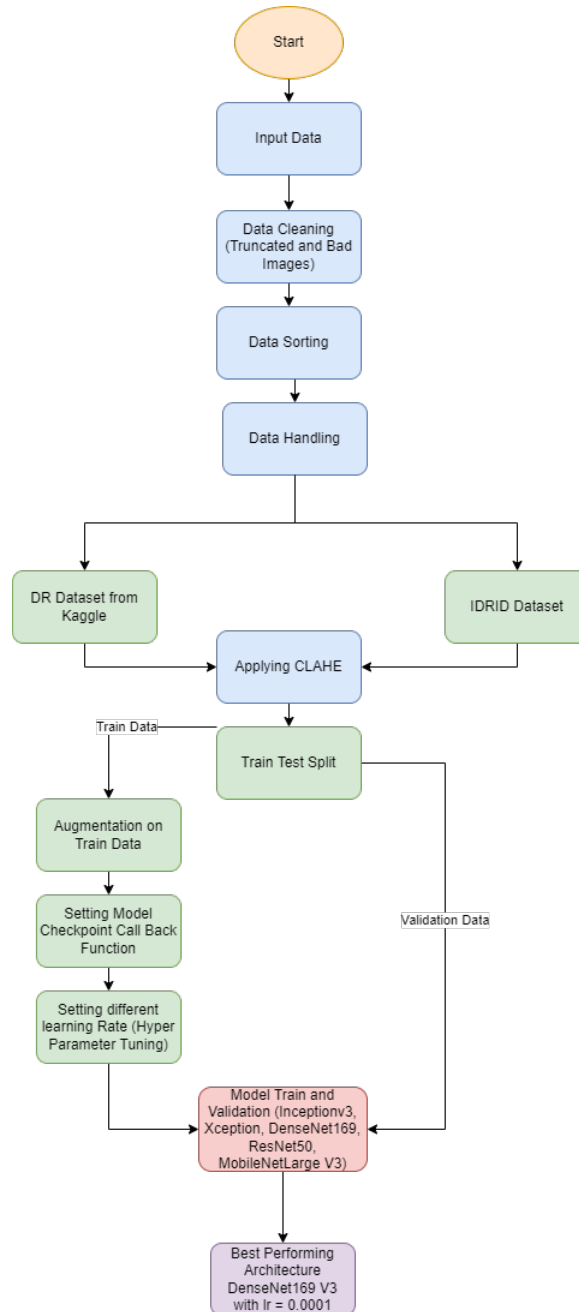


Figure 3.1: Workflow Diagram

3.1 Data Pre-Processing

Pre-processing entails a number of processes, as previously noted. We only have photos in our dataset. In order to feed our model with varying levels of diabetic retinopathy afflicted photos from two different dataset, we split them into distinct folders.

We organized the photographs into various folders based on the ailment and its severity degree using the provided labeled CSV. This allowed us to quickly determine the amount of data for each instance, as well as detect any potential data imbalances.

3.1.1 Data Cleaning

Image data is frequently skewed. For a variety of causes, images might become damaged, clipped, deformed, or truncated. The accuracy and validity of our model will be harmed if such data is included in the train and test.

As a result, our initial course of action was to eliminate such erroneous data. The shortened data is the first flaw here. Truncation is defined as the process of shrinking something, removing the least significant digits, replacing an angle, or performing a similar action.

In image processing, truncation operates in a similar way. When we try to process these photos, we usually get an error message like "OSError: image file is truncated!"

When we go through our data and come across an error like this, we may delete or transfer the image to a new location that we won't use in our code or model. Following the removal of shortened photos, we must also remove any photographs that are undesirable. In our scenario, we're talking about visuals that aren't readable to the model. Images were taken by several cameras in our diabetic retinopathy dataset. There were some photographs that were burned out, some that were fuzzy, and some that displayed surprising features for such a wide range of image quality. We manually eliminated those photos to produce a better model. Figure 3.2 shows various cases of poor image quality. Such photos will seriously sabotage our model development and testing.

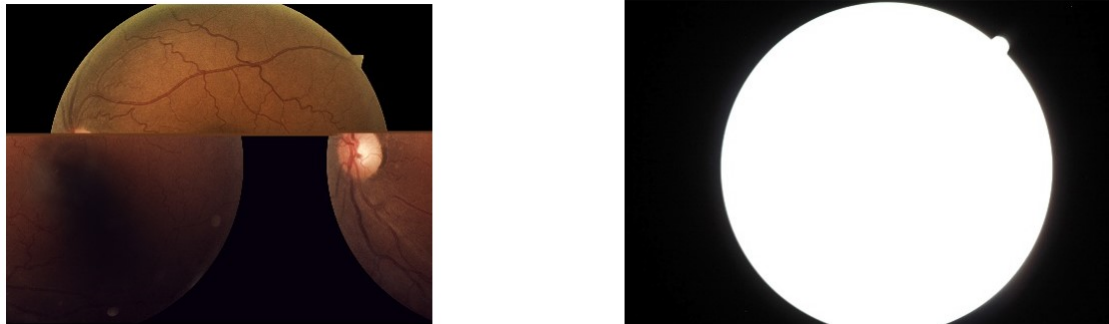


Figure 3.2: Truncated Image and Bad Image

3.1.2 Data Handling

Deep learning models are affected by data imbalance. The test accuracy will suffer a significant decrease if there is not enough data at each level on which a model is being trained. We used two Kaggle datasets in our project. In Fig. 5, we can observe that the dataset for Diabetic Retinopathy published by APTOS has a lot of imbalance in it. In all, we had 25624 photos for No DR, 2419 images for Mild DR, 5257 images for Moderate DR, 870 images for Severe DR, and 708 images for Proliferative DR. After cleaning the data, we had 708 images for Proliferative D.

However, the images of this dataset was captured in different light condition. For this reason, many of the images are washed out and burned. Some images were so bad that we had a difficulty of identifying it as a retinal scan. The use of a dataset with such a wide range of values results in a significant loss. We required to balance the dataset first in order to minimize the training loss, raise the validation loss, and enhance the validation accuracy.

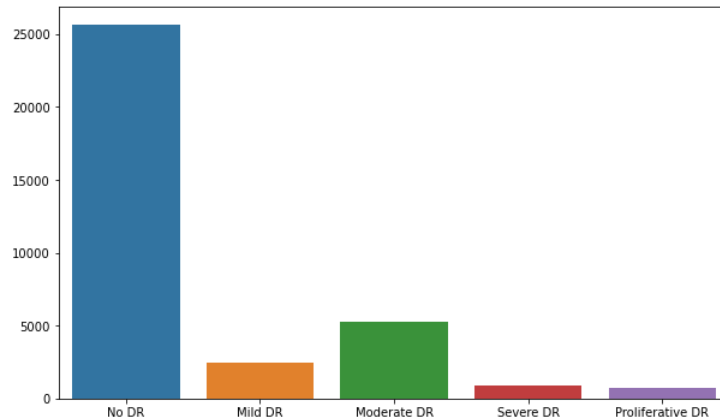


Figure 3.3: Imbalanced Data (APTOS)

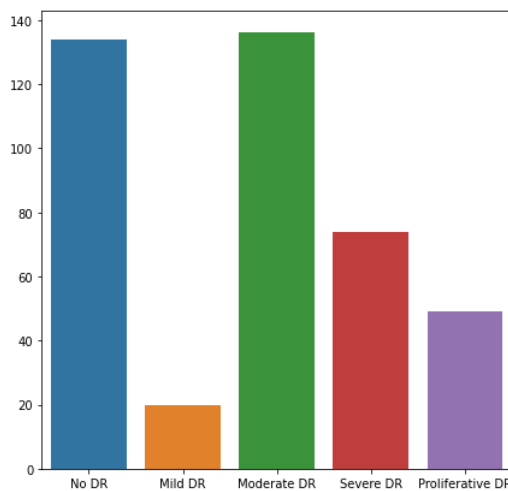


Figure 3.4: Imbalanced Data (IDRID)

The second dataset we worked with was Indian Diabetic Retinopathy Image Dataset [2]. The dataset was already splitted into training and validation sets. The training set contains total 413 Images and the validation set contains total number of 103 images. To be more specific, the training image contains 134 images for No DR, Only 20 images for MILD DR, 136 images for Moderate DR, 74 for Sever DR and lastly 49 for Proliferative DR. Moreover, this dataset is also imbalanced and a very small dataset. As the classification is defined by very small features, this small dataset will definitely cause underfitting. However, if we use deeper models for classification, there is a high chance that it will cause overfitting. But the good

part of this dataset is, the images are clean. There was not a single piece of bad image. All the images are clear and all the images have a good detail level.

However, we continue to have a dataset that is severely skewed. The number of samples collected for levels 1 to 4 is much lower than the number of samples collected for No DR in both of the dataset. level 4 and level 2 DR are very much less in quantity in both dataset. We could not use any of the dataset as it will give us a biased prediction. However, for bad images in our first dataset, the model will learn features which are not at all important for our research. But the challenge for us to get a good balanced dataset. Where not a single image should be bad or have unnecessary features like noise, shadows etc. The only possible option that came across our mind was mixing the two dataset and make a new one. This will improve the outcome of our model as it will not contain bad images and will be balanced.

3.1.3 Making Hybrid Dataset

To make a Hybrid Dataset, we decided to take all the images from IDRID dataset. Because, the images of this dataset are comparatively better than the other one. So, initially, we started with the total number of 134 images of level 0, 20 images of level 1, 136 images of level 2, 74 images of level 3 and 64 images of level 4. As we have many images which are in bad condition in the first Dataset from kaggle, we couldn't use a randomizer to pick images from that dataset randomly. So, we decided to hand pick some clean images and mix it with the IDRID dataset. So we picked images from the first dataset very carefully one by one. We were really careful about taking images which has the less amount of noise. We took 367 images of level 0 from the first dataset and mixed it with the level 0 of the IDRID dataset. Additionally, we took 473 images of level 1, 371 images of level 2, 435 images of level 3, 458 images of level 4. Finally after mixing two dataset, we now have 501 images of level 0, 493 images of level 1, 507 images of level 2, 509 images of level 3, 507 images of level 4. Now, if we take a look at (fig. 3.6), we can see that we have a pretty much well balanced dataset than before. As we have handpicked the data, we do not need to worry about bad images with noise such as blur, bokeh, shadow etc. Now it makes, we have a total number of 2,517 images in our training dataset to work with.

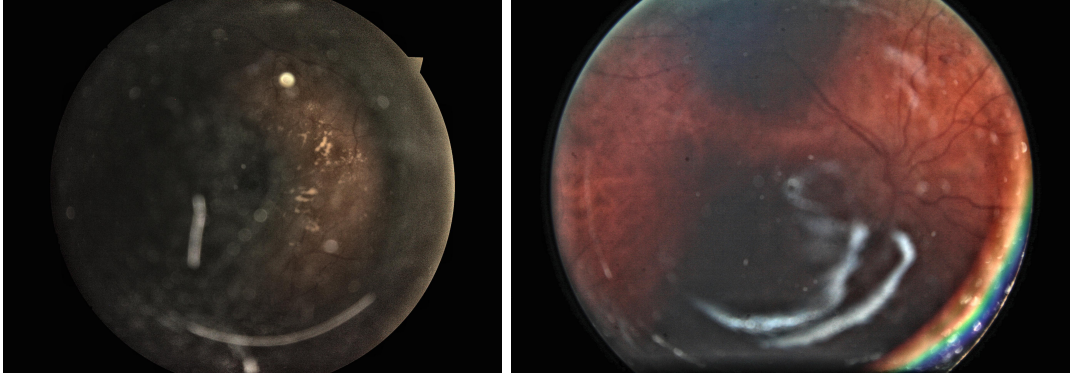


Figure 3.5: Bad Images

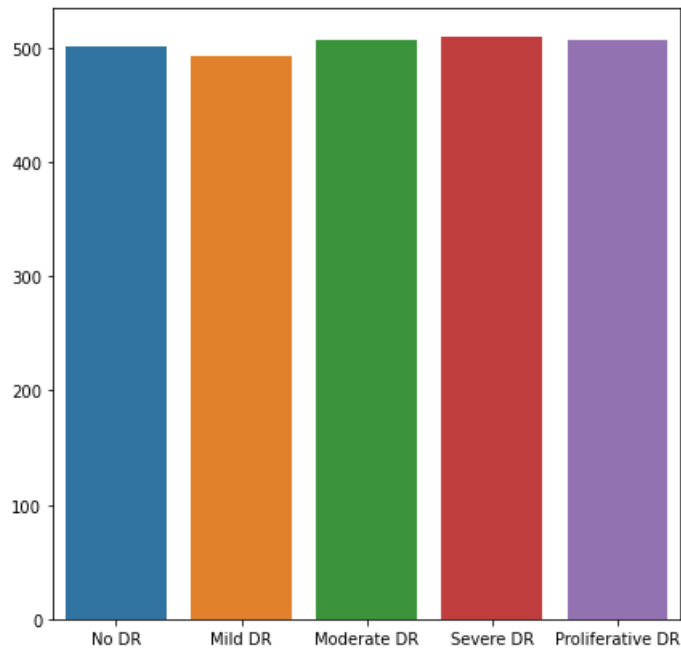


Figure 3.6: Hybrid Dataset Count Visualized

3.1.4 Image Enhancement using Contrast Limited Adaptive Histogram Equalization (CLAHE)

In our data pre-processing, CLAHE is an essential component of our workflow. CLAHE, which is an abbreviation for contrast limited adaptive histogram equalization, is a method for increasing the contrast of target pictures in order to improve visibility in low light.

It is possible to decrease issues such as noise amplification by the use of CLAHE. The picture is processed in this manner by working on little chunks of it rather than on the entire image as a whole. Iteratively combining the bi-linear interpolation of surrounding tiles is performed on each tile in order to eliminate any artificial boundaries from the image.

A far better set of results is obtained by applying CLAHE on the luminance channel of an image (HSV, hue-saturation-value) rather than applying the same technique

to the RGB channels of the picture.

Blood vessels, which are visible on retinal pictures, are a prominent aspect of these images, which we must consider while dealing with them. Microaneurysms (MA), which are red dots in the blood vessel, can be found in any stage of diabetic retinopathy. Additionally, MA is important information since it can diagnose DR at an early stage in the disease's progression. A good contrast in the blood vessel is therefore essential for this reason as well. In order to produce contrast in a picture, the intensity value range must be combined with the highest and minimum pixel values to be distinguished. The purpose of improving a picture with histogram manipulation is to obtain an intensity distribution that is uniform over the whole image field. A low effective intensity range is associated with images that have a low contrast. When using histogram equalization, you may spread out the intensity distribution while also adjusting the intensity of the original image. In accordance with [20], one of the distinctive characteristics of retinal pictures is that they include significant information in the G (green) channel. CLAHE will be applied to the G channel of our dataset for the objectives of our research with the expectation that a better result will be obtained.

Figure 3.7 shows two photos that illustrate the differences between CLAHE and other systems.

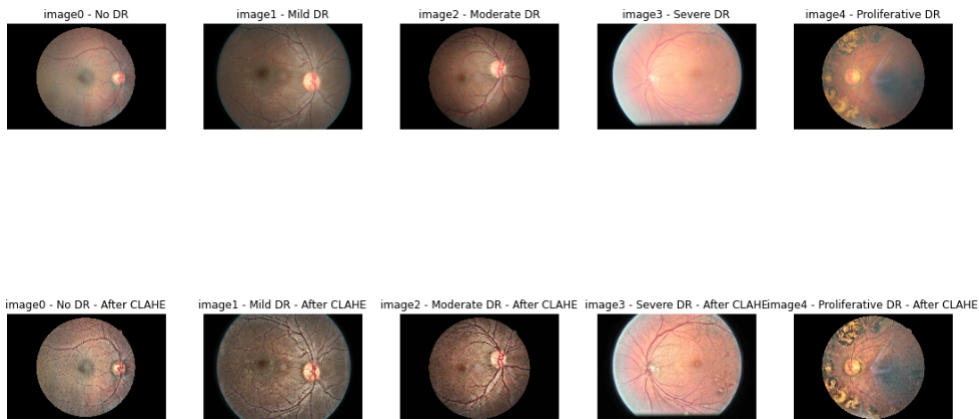


Figure 3.7: Retinal Images before and after using CLAHE

3.1.5 Augmentation

After balancing the data, we get a large number of photographs that are comparable. This is due to the fact that we have raised the sample count for Mild DR, Severe DR, and proliferative DR numerous times. As an added bonus, we boosted the sample count for the Moderate-Difficulty DR by a factor of 0.35. Because there are so many comparable photographs, the model will be biased. To put it another way, it will only remember a specific sort of picture and will predict it otherwise if the provided test image is a different type of image from the one remembered. It will only remember the image that we will show it when we train it. A bias model will have a significant impact on the correctness of our validation. In order to eliminate this prejudice, we applied augmentation techniques. For data pre-processing, we

made use of the ImageDataGenerator package provided by the TensorFlow Keras framework. This gadget is simple to use and quite effective. Depending on the parameter, it spins, shears, flips, and performs a variety of other operations on the photos to generate a new image. Because we have a large number of photographs that are identical, this program will make them unique by applying random techniques to them.

3.2 Convolutional Neural Networks

CNN or Convolutional Neural Networks are a form of deep learning methods. These methods or models assign priority to various aspects in the input images. Learnable weights and biases are the key cogs that help in building and making the model understand what it needs to recognize. Convolutional neural networks (CNNs) are sophisticated image processing AI systems that employ deep learning to handle both generative and descriptive tasks. A neural network is a hardware and/or software system modeled after the way neurons in the human brain operate. A CNN employs a technology similar to a multi-layer perceptron that is optimized for low processing requirements. Through the use of appropriate filters, it is able to properly capture spatial and temporal relationships in a picture [21]. Due to the reduced number of parameters involved and the reusability of weights, the architecture performs superior fitting to the picture dataset. In other words, the network may be trained to better recognize the image's complexity.

For classification and computer vision applications, convolutional neural networks (ConvNets or CNNs) are more commonly used. They're made up of node layers, each of which has an input layer, one or more hidden levels, and an output layer. Each node is connected to the others and has a weight and threshold assigned to it. If a node's output exceeds a certain threshold value, the node is activated, and data is sent to the next tier of the network. The convolutional layer is the most important component of a CNN since it is where the majority of the processing takes place.

It requires input data, a filter, and a feature map, among other things. Let's assume the input is a color picture composed of a 3D matrix of pixels. The size of the receptive field is determined by the filter size, which is usually a 3x3 matrix [22]. The output array does not need to map each input value directly.

It simply has to be connected to the receptive field, which is where the filter is applied. This characteristic can also be described as local connectivity. Before starting the neural network training, there are three hyper-parameters that determine the output volume size that must be established. These include: 1. The number of filters. 2. Stride is the distance, or number of pixels, that the kernel moves over the input matrix, and 3. Padding.

The feature detector's weight values remain constant as it advances over the picture, a technique known as parameter sharing. A CNN adds a Rectified Linear Unit (ReLU) adjustment to the feature map after each convolution operation, imparting

non-linearity to the model. However, because subsequent layers may identify pixels within previous layers' receptive fields, the CNN's structure can become hierarchical [23]. Finally, the convolutional layer converts the picture to numerical values that the neural network can evaluate and extract patterns from. Let's assume we're trying to figure out if an image contains a bicycle. A bicycle is made up of several components such as a frame, handlebars, wheels, pedals, and so on. Within the CNN, the combination of its elements indicates a higher-level pattern, resulting in a feature hierarchy.

In a simple Convolutional Neural Network, we start with an input image and feed it through the network to get a predicted label output in a very simple fashion, as seen in the diagram below. To elaborate, the algorithm gives weights or significance, as well as biases, to specific elements and attributes of the input in order to categorize the test picture.

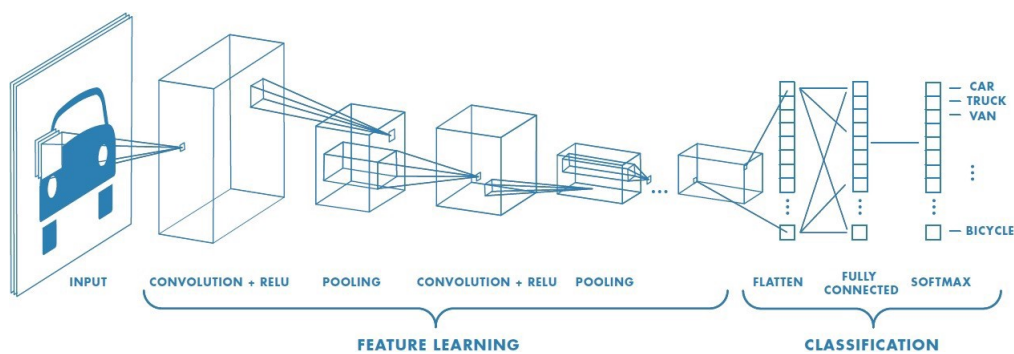


Figure 3.8: Convolutional Neural Network

Except for the first convolutional layer, which gets the input image, each consecutive convolutional layer consumes the output of the previous layer and creates an output feature map, which is then passed on to the next convolutional layer. Between the L number of layers, there are L direct connections - one for each layer and its next layer.

ConvNet, or Convolutional Neural Network, breaks down input pictures into little pieces or bits, making them much easier to analyze than the complete image while maintaining all of the image's characteristics. This enables a proper train-test with characteristics that are necessary for the model to successfully forecast. To illustrate how this picture segmentation works and how the kernel iterates over the segments, imagine that the whole image is partitioned into multiple little cubes, each of which is processed independently of their neighbors.

3.3 Model Training

We used ImageDataGenerator to divide the data one more before training the model, and we did so before training the model. We divided the data into two groups: 80 percent and 20 percent. 80 percent of the data is used for training, while the remaining 20 percent is used for testing. Splitting the data in this manner makes it easier to train the model and execute the necessary tests and validations after it is trained.

3.3.1 ResNet50

A deep convolutional neural network is a multilayer network that naturally combines low, mid, and high level features and classifiers in an end-to-end multilayer manner. The features can be increased by increasing the depth of the network. Deep convolutional neural networks are used to classify images. However, it has now been demonstrated that deeper networks can result in a significant gain in accuracy, but that this rise is short-lived after a few rounds. This problem was addressed by a group of researchers who developed a deep residual learning architecture that was eventually dubbed ResNet as a solution. The layers of this architecture are designed to suit a residual mapping. The intended underlying mapping was labeled as $H(x)$, and the stacked nonlinear layers fit another mapping of $F(x) := H(x)x$, which was denoted as $F(x)$. In the original mapping, $F(x)+x$ is substituted for $F(x)$. As previously stated, it has been demonstrated that optimizing a residual mapping is more easier than optimizing the old conventional mapping [24].

ResNet50 is one of the few CNN designs that is capable of continually delivering good accuracy despite having a large number of nodes in the network. The fundamental premise of this design is centered on the use of shortcut connections or residuals to create deeper models rather than using full models. ResNet has used batch normalization as a standard practice. It is capable of handling 26 million parameters [24]. The design is capable of working with up to 152 layers while still delivering great outcomes.

By simply looking at the basic diagram (fig. 3.9) shown below, it is possible to quickly visualize the building components of the ResNet design.

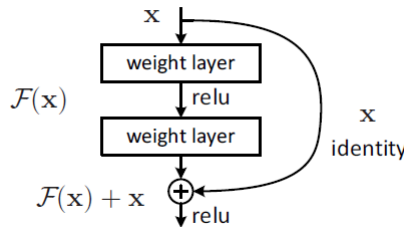


Figure 3.9: Residual Learning: Building Block

3.3.2 Inception or GoogLeNet

The architecture of Inception may be summarized by stating that it is a CNN (convolutional neural network) with a depth of 27 layers, which is the shortest description possible [25]. The Inception model's goal is to eliminate reliance on fully linked network designs by replacing them with sparsely connected network architectures, therefore removing the need for them.

The following graphic depicts a comprehensive visual representation of the whole 27 layers of the work being shown (fig. 3.10). Following that, a more in-depth explanation of the operating mechanism is provided.

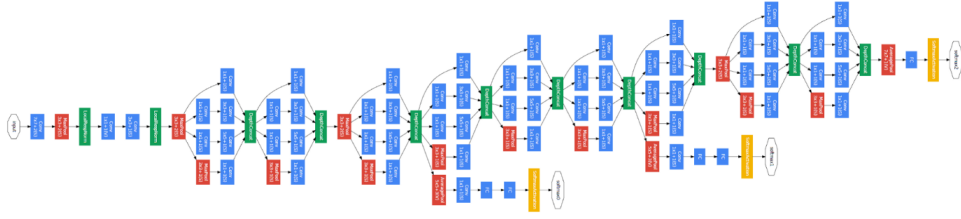


Figure 3.10: Layers of InceptionV3

It is possible to simplify the graphic above in order to make Inception more understandable. [26] The inception layer concatenates numerous convolutional layers, such as 1x1, 3x3, and 5x5, into a single output vector by combining them into a single output vector. In the following stage, this vector is utilized as the input for the next stage. The following is an illustration of how this section should be visualized.

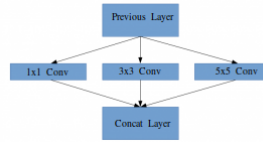


Figure 3.11: Simplified InceptionV3

Additionally, there is more to the Inception layer than what is seen above. These add-ons form a significant component of the Inception architecture.

3.3.3 MobileNetV3 Large

MobileNet is a CNN architecture model built for mobile and embedded vision. MobileNet stands out from other models since it requires relatively little computational power to execute or apply transfer learning. This makes it ideal for mobile devices, embedded systems, and computers that lack a GPU or have low processing efficiency without sacrificing considerable accuracy. It's also ideal for web browsers, which have limitations in terms of compute, graphics processing, and storage. In 2017, the first version of MobileNets was published. The main goal was to present a set of TensorFlow-based computer vision models that maximize accuracy while also taking into account the limited resources available for an on-device or embedded application. To reduce the amount of parameters, MobileNetV1 incorporated depthwise convolution [27]. The second iteration adds an expansion layer to the block, resulting in a three-layer expansion-filtering-compression scheme. This approach, dubbed “Inverted Residual Block”, helped to improve performance even more. MobileNet V3 is a review of MobileNet V1 that was presented at ICCV in Seoul, South Korea in 2019. It tries to optimize the number of filters for every conv in question and picks the model with the highest across-the-board algorithm. This version adds Squeeze and excitation layers to the initial building block taken from V2.

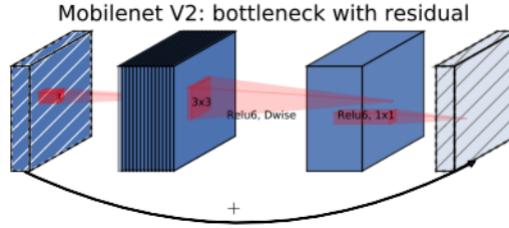


Figure 3.12: MobileNetV2: Bottleneck with residual

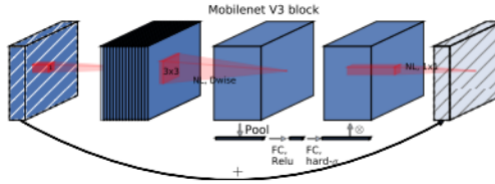


Figure 3.13: MobileNetV3 Block

MobileNet V3 has two structures: MobileNetV3-Large and MobileNetV3-Small. Both have the following structure (MobileNetV3-Large on the left and MobileNetV3-Small on the right):

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Table 1. Specification for MobileNetV3-Large. SE denotes whether there is a Squeeze-And-Excite in that block. NL denotes the type of nonlinearity used. Here, HS denotes h-swish and RE denotes ReLU. NBN denotes no batch normalization. s denotes stride.

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

Table 2. Specification for MobileNetV3-Small. See table 1 for notation.

Figure 3.14: MobileNetV3 Large Structure and MobileNetV3 Small Structure

3.3.4 Xception

Xception is a Depthwise Separable Convolutions-based deep convolutional neural network architecture. Francois Chollet, a Google Inc. employee, introduced this network. Xception is a "extreme" version of an Inception module that performs numerous transformations (as seen in the diagram above) on the same input. Xception stands for "extreme inception," and it pushes Inception's concepts to their logical conclusion. 1x1 convolutions were employed to compress the original input in Inception, and different types of filters were utilized on each of the depth spaces from each of those input spaces [28]. Xception simply reverses this process, applying the

filters on a depth map before compressing the input space using 1X1 convolution applied across the depth. This approach is virtually equivalent to a depthwise separable convolution, which has been used in neural network building since 2014. The occurrence of non-linearity after the initial operation is another distinction between Inception and Xception. In the Inception model, both processes are followed by a ReLU non-linearity; however, Xception does not add any non-linearity.

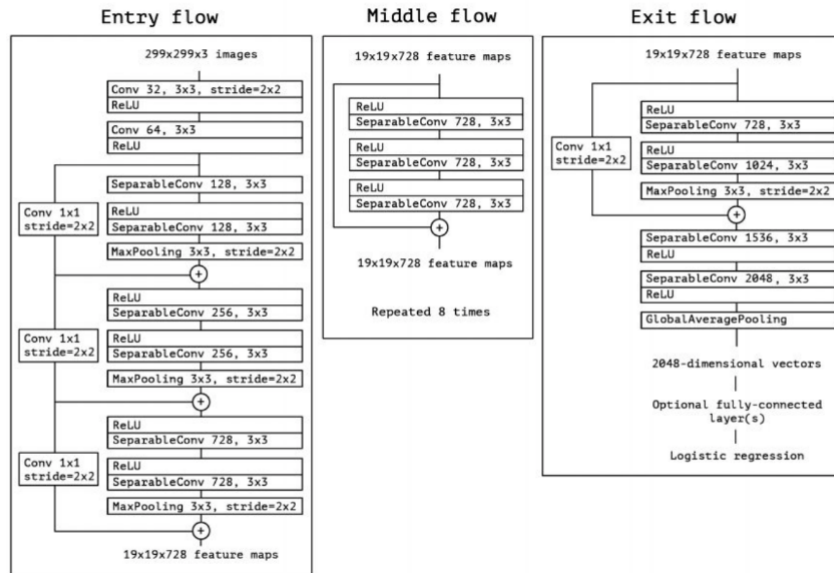


Figure 3.15: Xception Workflow

Xception outperforms every model in Imagenet Dataset:

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	0.790	0.945

Figure 3.16: Xception vs Other Models in Imagenet Dataset

The validation accuracy of the Xception model is also higher than that of the inception model presented below

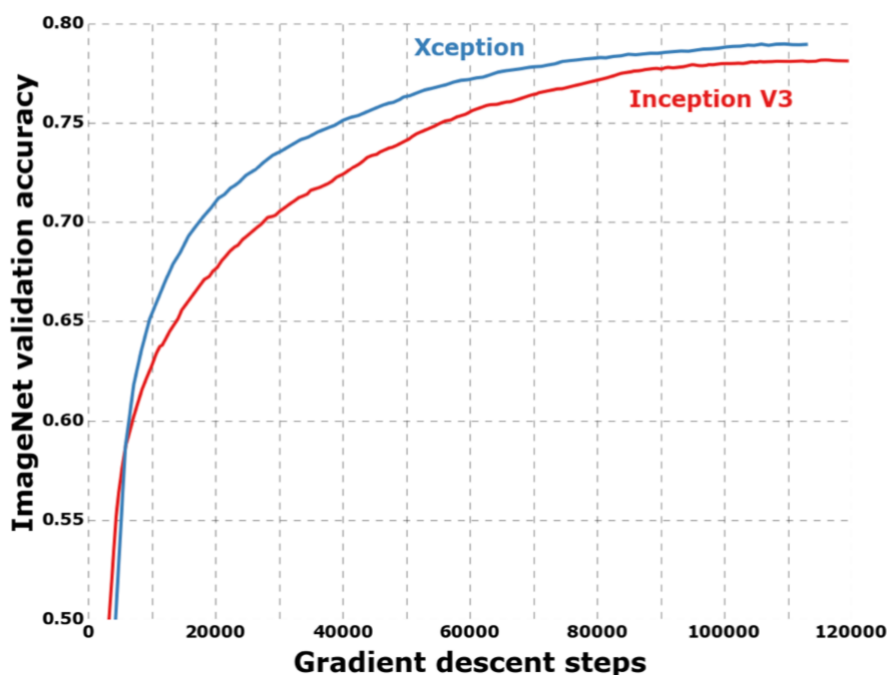


Figure 3.17: Xception vs Inception V3

3.3.5 DenseNet-169

Densely Connected Convolutional Networks (DenseNets) are the next step in the evolution of deep convolutional networks' depth. This is due to the fact that the path for information from the input layer to the output layer (as well as the gradient in the other way) is so long that it is possible for them to evaporate before reaching the other side. To do so, they simply connect each layer to the next directly. DenseNets are extreme deep neural networks in which each layer has direct access to the loss function gradients and the original input image. Transition Layers are the layers between them that handle downsampling by using batch normalization, 1x1 convolution, and 2x2 pooling layers [29]. DenseNet comes in a number of different configurations, including the DenseNet-121, DenseNet-160, DenseNet-201, and others. The numerical numbers correspond to the number of layers in a neural network. DenseNet is separated into DenseBlocks, each of which has a different number of filters but the same dimensions inside the block. The Transition Layer uses downsampling to provide batch normalization; it is a crucial stage in CNN. The number of filters varies amongst DenseBlocks, with the number of filters increasing the channel size. The l -th layer's generalization is aided by the growth rate (k). It's in charge of calculating how much data should be added to each layer.

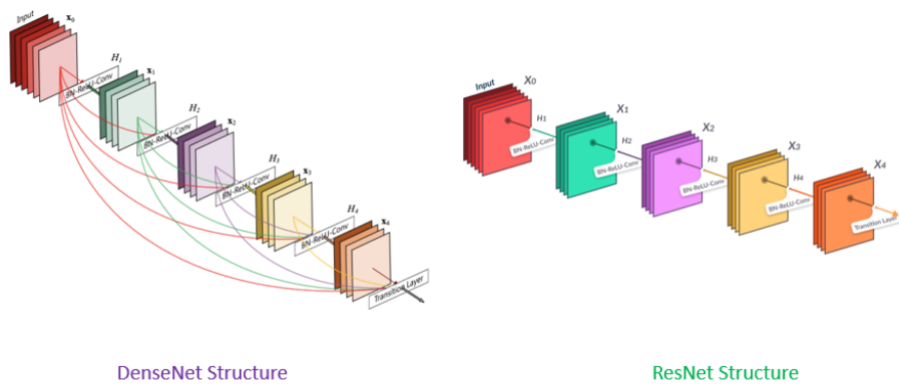


Figure 3.18: 5-layer dense block with a growth rate of $k = 4$ and the standard ResNet structure

3.4 Adamax Optimizer

Adamax is an advanced optimizer variant enforcing the Adamax algorithm. This optimizer is takes into account the infinity norm base of Adam. Individual weights in Adam are updated according to a rule that scales their gradients inversely proportionate to a (scaled) L^2 norm of their individual current and previous gradients. In order to generalize the L^p norm based update rule, we must first define the L^p norm in the first place. When p is large enough, such versions become numerically unstable [30]. However, in the exceptional scenario where, let p be a positive integer, a remarkably simple and reliable solution is discovered.

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Figure 3.19: Adamax Formula

3.5 Hyper-Parameter Tuning

Hyper-parameters shorten training time in half and boost performance. Hyper-parameters also influence how an algorithm functions and have an impact on the final result. It stores the data that governs the training process. For this hyper-parameter, settings are selected randomly from a user-defined search area. Following that, they are identified based on the re-sampling chosen approach. For unknown hyper-parameter values, the regression model evaluates the performance of the machine learning algorithm. The tuning process itself is generally influenced by individual and intuition. When compared to approaches that do not employ knowledge from prior runs, such as random search or grid, this tuning strategy drastically reduces the tuning budget needed to discover a setting that is near to the global minimum. The test/validation loss may be utilized to get insight into the training process, while the final test accuracy can be used to compare results. The search for hyper-parameters can be significantly accelerated if the hyper-parameters are specified

using only a few epochs. Without doing a full training to compare the final performance results, the test loss throughout the training process may be utilized to determine the ideal network design and hyper-parameters [31]. Here is a summary of the effects of each hyper-parameter.

Learning Rate

It performs a LR range test with a small learning rate which increases slowly in linear order throughout a pre-training run and finds the maximum learning rate. It also performs Cyclical learning rates to choose the learning rate. There is a maximum pace at which the learning rate can increase without the training becoming unstable, which has an impact on your minimum and maximum learning rates options.

Total Batch Size

A big batch size works effectively, although the magnitude is usually limited by GPU memory. Larger batch sizes are favored too much by a fixed number of iterations. The reason is that with the 1 cycle learning rate schedule, greater batch sizes allow for higher learning rates.

Momentum

It is related to Learning rate. It's important to establish as much momentum as possible during training without producing instabilities, just as it is with learning rates. If a constant learning rate is applied, a big constant momentum will speed up the training and enhance the learning rate. Short tests with momentum values of 0.99, 0.97, 0.95, and 0.9 will reveal the ideal momentum value rapidly.

Weight Decay

At this part the values must be set properly. This needs a grid search to identify the right magnitude, but usually only one significant figure accuracy is required. Smaller datasets and architectures appear to necessitate higher weight decay values, whereas larger datasets and deeper architectures appear to necessitate lower values

Chapter 4

Implementation

4.1 Dataset

4.1.1 Source

The datasets utilized in our research, as well as the proposed model, were obtained from Kaggle and IEEE journal. One of the dataset was published by APTOS on Kaggle and the other one was published in IEEE on 2018 [1], [2]. Additionally, links are provided in the bibliography section and are also referenced further down in this section. Two valid Diabetic Retinopathy datasets were utilized in conjunction with one another to create a hybrid dataset for our needs.

4.1.2 Dataset Description

As we are working on identifying diabetic retinopathy, we have discovered that it can only be detected by the retinal scans taken from the patients themselves. We have collected two datasets. The first one which was published by APTOS on kaggle had in total 88,702 images [1]. Among them, there are 35,126 images for the training sets. The images of this dataset was captured in different light condition. The dataset was published for a competition on 2015. This dataset is also the largest retinal scan dataset available. This dataset has 25624 photos for No DR, 2419 images for Mild DR, 5257 images for Moderate DR, 870 images for Severe DR, and 708 images for Proliferative DR. After cleaning the data, we had 708 images for Proliferative D. All the images are in different shapes. some are in 3000x3000 pixels or some are even bigger or smaller.

The second dataset we are working on is Indian Diabetic Retinopathy Image Dataset [2]. This dataset was published on IEEE in 2018. This dataset contains total 516 images. Among them, 413 in training set and 103. This dataset represents the Indian population. Moreover, only this dataset contains the annotation of different lesions at a pixel level. This dataset was divided into 3 from after some pre-processing. 1. Segmentation , 2. Disease Grading, 3. Localicization. For the purpose of our research, Original images from segmentation were enough, so we worked on the original images .In the original images, we had 134 images for No DR, Only 20 images for MILD DR, 136 images for Moderate DR, 74 for Sever DR and lastly 49

for Proliferative DR just in the train set. This is comparatively a small dataset than the first one we chose, but this dataset was better in terms of image quality than the first one. All the images are in a constant shape which is 4288x2848.

4.1.3 Data Sample

Obtaining fundoscopy pictures is a difficult and time-consuming process. We contacted a number of hospitals and clinics in the hopes of acquiring annotated fundoscopic photos from an expert, but unfortunately, our luck did not shine on us this time.

The problems associated with getting such medical information have been discussed in earlier articles in the same topic. This pattern appeared to be extremely widespread in many of the articles that we read for our research, and we were able to identify it rather easily. Because of this, we turned to Kaggle in order to collect datasets that we could utilize to train and test our model. Because none of the datasets were complete in themselves, we aggregated several datasets of equal size and quality to conduct our research. Detailed information on the datasets we utilized may be found in the reference section [1], [2].

Alternatively, we have included some samples of the data we have used to train and test our model in the following figure (fig. 4.1).

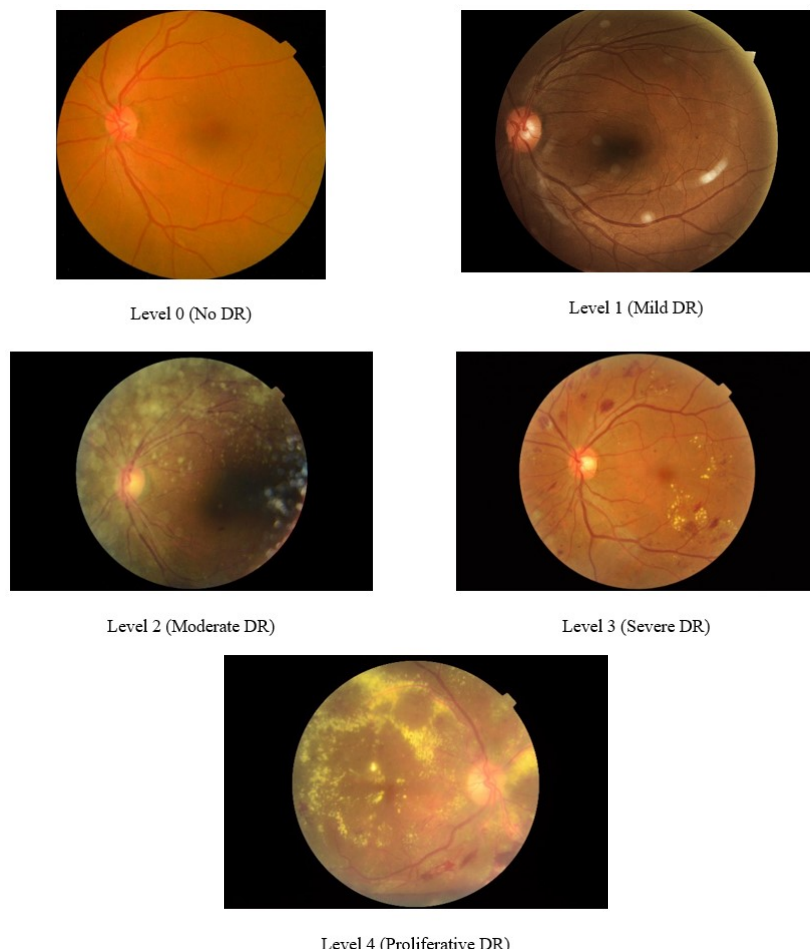


Figure 4.1: Sample Data

The phases of the sample analysis operations are carried out by trained doctors or non-eye-trained workers based on sensitivity and specificity findings which are equivalent to or better than human expert observations. The overall performance of automated software for this diabetic retinopathy diagnosis is high, with a Sensitivity of more than 80% and a Specificity of more than 90% [32]. Diabetic Retinopathy (DR) is a diabetic complication that causes the retina's blood vessels to widen and leak fluids and blood [33]. The presence of various sorts of lesions on a retina picture is used to identify DR. Microaneurysms (MA), haemorrhages (HM), and soft and hard exudates (EX) are examples of these lesions (fig. 4.3-4.6).

Microaneurysms

Microaneurysms show up as tiny red spots. MA was divided into six categories by Michael et al [34].

Haemorrhages

Flame (superficial HM) and blot (deep HM) are the two forms of haemorrhage.

Hard Exudates

It looks like bright-yellow patches on the retina triggered by plasma leakage. They are located in the retina's outer layers and feature sharp borders.

Soft Exudates

Because of the enlargement of the nerve fiber, it appears as white spots on the retina. It is also known as cotton wool spots.

Depending on the occurrence of these lesions, there are five phases of DR as follows.

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

Level 0 - No DR

Here we can see the absence of lesions. A sample fundoscopic image with no traces of Diabetic Retinopathy is given below to help in visualization.



Level 0 (No DR)

Figure 4.2: Level - 0: No DR

Level 1 - Mild DR

The initial symptom of DR is microaneurysms (MA), which appear as little red spherical spots on the retina due to a weakening in the vessel's walls. At this point, small quantities of fluid may flow into the retina, causing the macula to enlarge. We can also notice sharper borders with size size that is less than 125m.

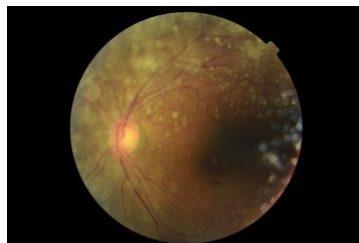


Level 1 (Mild DR)

Figure 4.3: Level - 1: Mild DR

Level 2 - Moderate DR

It's more than merely microaneurysms. Swelling of tiny blood vessels begins to obstruct blood supply to the retina, inhibiting normal feeding (fig. 4.4). Bigger bright-yellow spots are seen on the retina which are more than 125 m and have an asymmetrical edge.



Level 2 (Moderate DR)

Figure 4.4: Level - 2: Moderate DR

Level 3 - Severe DR

A bigger part of the retina's blood vessels becomes clogged, resulting in a considerable reduction in blood flow to this area. Exudates occur in the macular area, and the macula thickens.

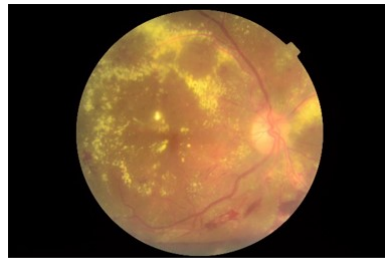


Level 3 (Severe DR)

Figure 4.5: Level - 3: Severe DR

Level 4 - Proliferative DR

In this level, there are more severe and extensive mutations in the retina. Lens deterioration happens. This is where the retina pulls away from the back of the eye. At this stage, there is high risk of losing vision.



Level 4 (Proliferative DR)

Figure 4.6: Level - 4: Proliferative DR

4.2 Applying Deep Learning Algorithms

The suggested system will consist of one ipynb file and one .h5 file, both of which will include a stored version of the model in a different format. The model is put into action in a localized area. For our implementation, we made use of the Jupyter Notebook. It is equipped with an Intel i5 9th generation CPU, 16 GB of RAM, and an Nvidia 8 GB graphics card, which we employed in this experiment.

Our deep learning frameworks were written in Python, which provides a large number of deep learning frameworks that are extremely straightforward to use. Making deep learning models is also a common use of Python, which is readily available. TensorFlow Keras was utilized as the framework for this project. Furthermore, we employed the most recent version of TensorFlow, which is TensorFlow 2.5.0 with a GPU-based runtime, to achieve our results. We proceeded through five phases in the process of putting our suggested approach into action.

- Sorting and Data cleaning
- Balancing the dataset and applying augmentation
- Building a Hybrid Dataset

- Enhancing the images using CLAHE
- Feed data into different CNN architectures
- Analyze results

For the purposes of our research, we required a high-quality dataset that contained OCT scan images of normal eyes as well as images of eyes afflicted by DR. We feed the information into our CNN model with the goal of identifying illnesses and classifying whether the patient has DR or not, as well as at what stage the disease is currently at. We fed the data into different models. With the selected architectures, such as ResNet50, DenseNet169, MobileNetV3Large, Xception and Inception V3, we are able to categorize images according to the steps described in the dataset.

The Transfer Learning technique was utilized to improve the results obtained when using diverse architecture. The dimensions of the input data shape were 1024x1024x3. This implies they will be taking RGB color photos with three channels at a resolution of 1024x1024 pixels.

We used a number of different architectures for the purposes of our investigation. For example, ResNet50, DenseNet169, MobileNetV3Large, Xception and Inception V3. Transfer Learning is the method we will apply for this challenge. First, we removed the top layer of the design in order to implement the transfer learning technique. Afterwards, we set all of the trainable layers to false so that we could keep the trainable layers in place. After that, we manually added four layers to achieve a satisfactory result and classification, consisting of one Flatten layer and three Dense layers. The flatten layer's purpose was to flatten the output of the convolutional block that followed it. Finally, we added a categorization layer to the mix. We utilized the SoftMax activation function for the model that would classify DR because it is a multi-class classification model.

4.2.1 Train Test Split

In order to Train and Validate our model, we split our data in a 80:20 ratio. This is the regular splitting ratio to get the best performance evaluation of a Deep Learning model. The 80% of our data will be used for training. This portion of the dataset has all the features which distinguish the level of diabetic retinopathy. We will feed this data to our model. Validation set which is the other 20% of our data will be used to evaluate our model in terms of unseen data. Which means, the model will be shown some data which it did not see before. And it will predict the level of DR of that data. All the images of the validation set will be tested and the loss and accuracy we will get from the model will be an important metrics for our model evaluation.

4.2.2 Taking Image Input

As we have said in the upper section that all the data were stored in different folders according to the label. We did that because it is easier to read these images using ImageDataGenerator in this way.

ImageDataGenerator is library from Tensorflow which can flow the image from any directory and feed into the model. And it makes way easier and memory efficient if the dataset is large. ImageDataGenerator has various features such as augmentation. It also takes batch size as a parameter which we can manually control. By Controlling the batch size we can set the batch size in a level where the model does not throw a **Resource Exhausted Error** while training.

We, Used ImageDatagenerator and set the Rescale Parameter to 1./255. so that all the pixel values must be in between 0 to 1. Then we used the flow_from_Directory method fetch the data and set the input size to 1024,1024. We also set the batch_size 10 which means while training, it will take 10 images per step. We also set batch_size to 10 in terms of validation dataset. Also we set the class_mode to categorical because our dataset has multiple class.

```
Found 2511 images belonging to 5 classes.  
Steps Per Epoch: 252  
Found 504 images belonging to 5 classes.  
Validation Step: 51
```

Figure 4.7: Output of ImageDataGenerator

After taking input by using this method, we get total number of images and by a simple calculation we can also get the steps we need to set during the training of our models. (fig. 4.7)

4.2.3 Augmentation

Augmentation is used when we have a small dataset and the model is under fitting. We are working on a hybrid dataset which has around 2511 data on training set. Which is not that much less but not enough. To avoid underfitting, we used augmentation. The good thing is we used ImageDataGenerator for taking input from directory and ImageDataGenerator have a builtin feature for augmentation. It takes some parameters while declaring the object for Augmentation such as rotate, sheer etc. It randomly change the image based on the parameters and feed into the model. The parameters we have used for the augmentation have been discussed below.

- We used a rotation range of 30°, which means that a picture may be rotated in any direction between 0° and 30°.
- The width and height shift ranges were also employed, with the value set to 0.1 for each of the three parameters. These two options will randomly change the picture's width and height by 10 percent, resulting in a distorted image.
- We also used shearing to get the job done. Shearing is, in essence, a shift of point of view or viewpoint. We used it for 0.2 percent or 20 percent of the total.
- we utilized 20 percent of the zoom range, which would randomly zoom the photo between 0 and 20 percent of its original size.
- In addition, we utilized horizontal flip, which just flips the image horizontally rather than vertically.
- if the picture is manipulated, it may generate some additional pixels in order to keep its structure. As a result, we chose continuous fill mode, which will result in the creation of additional black pixels.

Following the use of these augmentation settings, it will randomly modify the image. Due to the fact that it added more detail to some of the retinal pictures, we decided not to employ closest mode since it may have an adverse effect on the model's performance.

4.2.4 Our CNN Model

We have used Transfer Learning technique for our model. Tensorflow keras offers transfer learning and it is pretty easy to implement. Keras has built in library from where we can directly use well known Architectures with the 'imagenet' weights. ImageNet is a dataset from a competition named Large Scale Visual Recognition Challenge (ILSVRC). This is a huge dataset. The challenge was to classify total 1000 classes from this dataset. Tensorflow Keras offers all the established Architecture for this competition and also the new architectures with the weights of this challenge. It makes those architectures more efficient and better in terms of classification.

The Architecture we used was DenseNet169. It is one of the unique architecture which does not feed forward like other regular Architectures. It feed the information differently to the next layer. This architecture introduced a new concept of flowing the information to the next layer. The connection between a layer and all the following layer were done in a way that all of the following layers are connected. So when a layer send information, it sends the information to all the following layers together. If we put it on another way, the n^{th} layer will send the processed information to all other following layers which is $[n+1, n+2, n+3+....]^{\text{th}}$ layers. This kind of dense connectivity makes this architecture unique and gives a good performance.

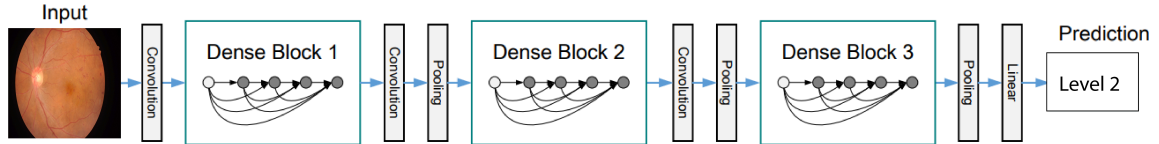


Figure 4.8: DenseNet169 Architecture

In our research we used DenseNet169 as it performed better than any other architectures. This architecture has total 6, 12, 12, 32, 32 CNN layers in four dense blocks. As we have a Medium sized dataset, we did not use DenseNet201 as it might overfit. We did not use DenseNet121 as it might underfit our data. We have done further analysis by comparing other few architectures and went through numerous trial and error processes. But for our dataset and research, DenseNet169 gives the best output.

Using the Keras Tensorflow API, we used the DenseNet169 built-in module. We set the input shape to $[1024, 1024] + [3]$, so that while applying the first CNN layer, the pixels at the border do not get ignored. After that we set the weights parameter as 'imagenet'. Which means, the model we are going to train will already have the weight of the ImageNet dataset. This will make the model more efficient. After that, we set the include_top parameter to False. So the top layers will be removed. Now, our model is almost done as we do not have top layers yet. For the top layer, we set a Dense layer with the SoftMax activation function. The reason behind is the model is going to classify multiple classes. The number of tensors were set to 5. Because the model is going to classify 5 classes which are level 0, level 1, level 2, level 3, and level 4.

4.2.5 Compiling the model

We compiled our model with the loss function named categorical cross-entropy as our model is going to classify multiple classes. We set our metrics to accuracy. We used the Adamax Optimizer which is comparatively rare while using CNN. In very few researches of Diabetic Retinopathy detection used the Adamax optimizer. Generally, almost everyone uses Adam or RMSprop optimizer in their CNN models. Surprisingly we got very good results using the Adamax optimizer than the Adam or RMSprop optimizer. We also tweaked the learning rate a bit. In other words, we tried Hyperparameter Tuning in terms of getting a good result and the outcome was great.

4.2.6 Optimizer and Hyper-Parameter Tuning

Hyper Parameter Tuning is basically tweaking the Hyper Parameters to get better result. Epochs, Learning rate, batch size etc. these are the hyper parameters. The default value of learning rate of our optimizer is 0.001. We tweaked the learning rate and trained multiple different models through trial and error. After multiple trial and error we get the best result when we lower the learning rate to 0.0001 from 0.001. This improve the validation loss a lot. We were getting very large validation loss but when we set the learning rate to 0.0001, we saw a massive change and improvement in the validation loss. Because, While training our data, our model was getting far from the global minima while using the default learning rate. So, while compiling our final model we used the Adamax optimizer with the learning rate of 0.0001.

4.2.7 Fitting the model

Our DenseNet169 model was ready with the Adamax optimizer which had the learning rate of 0.0001. While fitting the model, we input our train and validation data. We set the steps per epoch to 252. The calculation behind the steps per epoch was, (Total train Image/Batch Size) + 1. Which means $2511/10 = 251+1 = 252$. By doing the same calculation on validation data, we get the validation steps which is 36. So, we set the validation steps to 51. We trained our model for 20 epochs. While training, we saw that there was a slight up and down movement in the accuracy. So, For our final model, we set a call back function to get the best result. We used Model Checkpoint Call Back function.

4.2.8 Model Check Point Call Back Function

Model Check Point Call Back Function is used in Deep Learning while training a model. This callback function saves the weights of the model when the model gives the best result. In other words, after each epoch, this callback function tracks the best result and save the weights on that epoch. Then on the next epoch, it checks again that if the new result was better than the previous saved one. If the result is better, it saves the new weights of the best result and it continues tracking the results. By using this technique, we get the best trained model even if the accuracy and loss jumps up and down a bit. We set the monitor parameter of this function to val_accuracy. So that it will monitor the accuracy at the end of the each epoch. We also set the mode to Max. This means, the call back function will monitor the validation accuracy after the end of every epoch and it will save the weights when it gets the maximum validation accuracy.

4.2.9 Model Evaluation

Our Proposed Model gave validation accuracy of 0.8829 which is 88.29%. Moreover, we got the validation loss 0.7083.

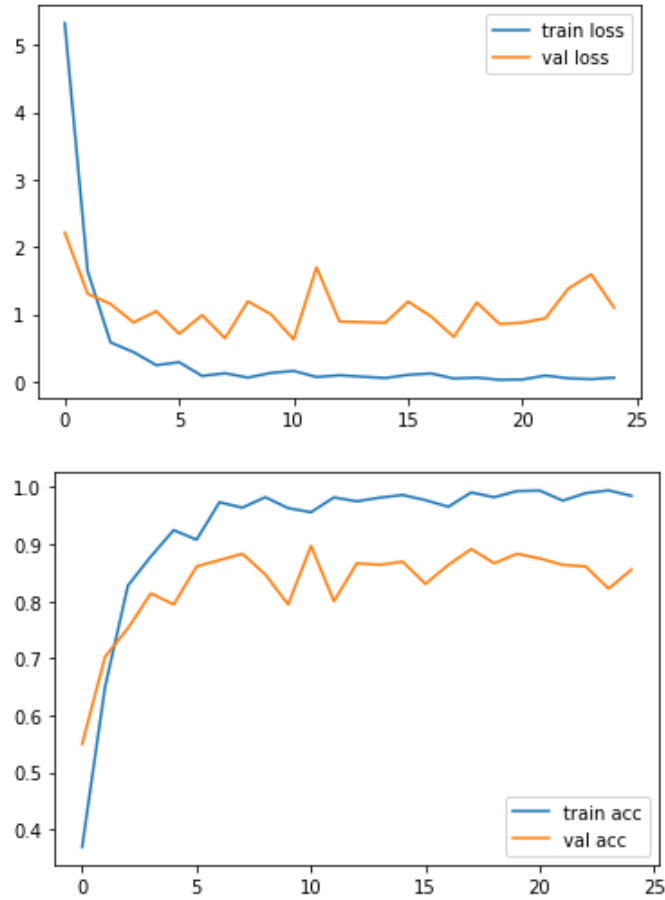


Figure 4.9: DenseNet169 Architecture Loss Accuracy Curve

If we observe our loss curve closely, we can see there's a gap between the both curve. This indicates Underfitting. Our model is definitely giving a good result. But it can be further improved. Basically, the core features for detecting DR are MA, HM, Hard Exudates, Soft Exudates are so small that 2511 images are not enough. We have large dataset from Kaggle but that dataset contains a lot of images which are full of noise. That means we faced a lack of clean data. So, if we can feed more good and clean data to our model, the probability of performing better is really high for our model.

The Confusion Matrix describes that how good the model is performing on our hybrid dataset. The Confusion matrix is given below.

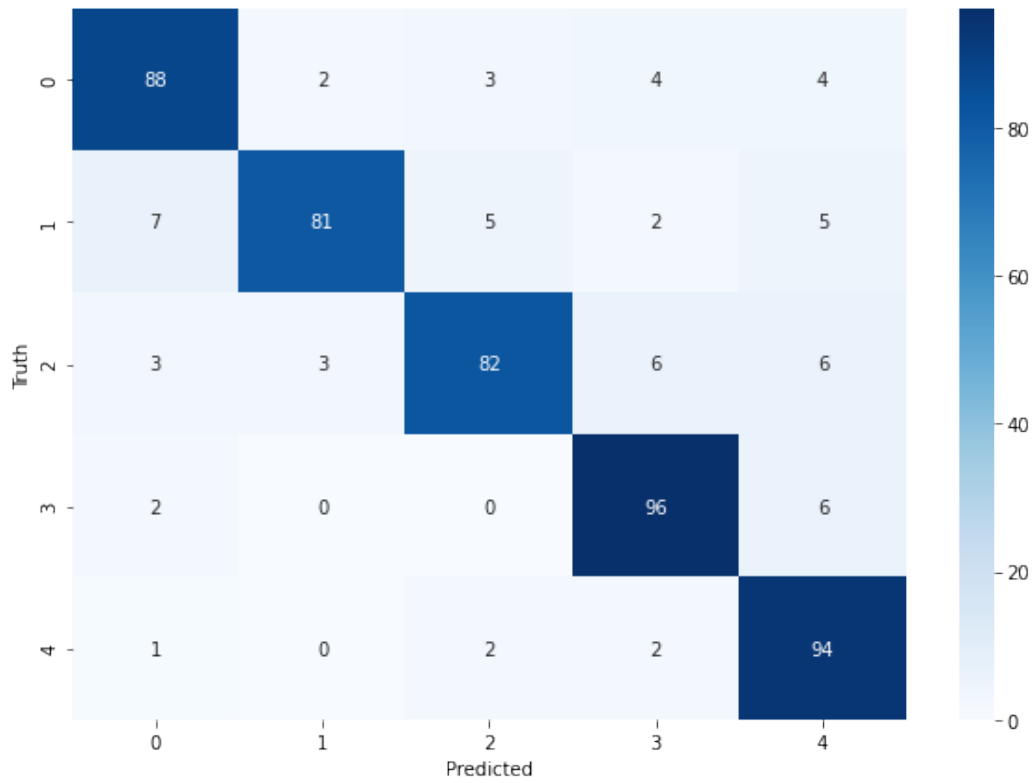


Figure 4.10: DenseNet169 model’s Confusion Matrix

And here is the classification report for our proposed model.

Classification Report					
	precision	recall	f1-score	support	
0	0.87	0.87	0.87	101	
1	0.94	0.81	0.87	100	
2	0.89	0.82	0.85	100	
3	0.87	0.92	0.90	104	
4	0.82	0.95	0.88	99	
accuracy			0.88	504	
macro avg	0.88	0.87	0.87	504	
weighted avg	0.88	0.88	0.87	504	

Figure 4.11: DenseNet169 model’s Classification Report

Chapter 5

Results and Analysis

5.1 Result Overview

The end result of our research is quite unique and we are astounded by the findings. To start off, there are a few terms and concepts that needs to be understood in order to properly comprehend the shortcomings as well as they final outcome.

5.2 Model Fit

The fitness of a model can be categorized into three distinct levels. Any model can be one of the following three types. A model deviates from perfect state due to various reasons that can be related to data pre-processing, training and even at testing. Underfit or overfit models will result in a very poor performance of the model.

- Underfit Model
- Overfit Model
- Optimum Model

5.2.1 Underfit Model

The term underfitting refers to a situation where a data model is unable to effectively represent the connection between input and output variables. This leads to a greater error margin not only on the training data but also the test or prediction data. An oversimplified model that did not have enough training time is one of the primary reasons of underfitting. If a model is trained with mostly precise and small-scaled features that are easy to miss and mistake for, that may also lead to underfitting. When a model is insufficiently generalizable to fresh data, that model is in no way suitable for prediction tasks. A models ability to correctly process new data and make accurate predictions is what enables us to utilize machine learning algorithms in classifying and making predictions.

Underfitting is indicated by a high bias and a low variance. Due to the fact that this behavior is visible when the training dataset is used, underfitted models can

typically be easily recognized than their significant counterparts, overfitted models. If we look at the train loss and validation accuracy of a model, we can easily distinguish if the model is underfitted or overfitted.

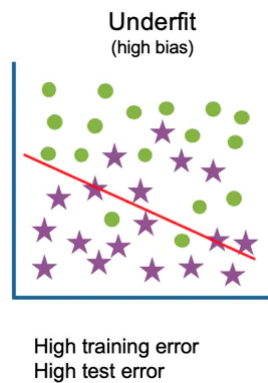


Figure 5.1: Underfit

5.2.2 Overfit Model

Overfitting notion in data science refers to scenario where training data and the model fits perfectly. This occurs when the developed algorithms are unable to accurately and properly process new data or test data ignoring the initial target. We all know that a machine learning algorithm or model is built with providing a decent portion of training sample data. However, when this train time exceeds the normal or if the model is sophisticated enough, it will start learning the irrelevant features or "noise" as they are called in these terms which refers to irrelevant or unwanted information. A model being able to recognize such noise and irrelevant features is said to be overfit and is incapable of correct predicting on data that it has not seen before.

Overfit model can be identified having low error margin and large variance. To bypass overfitting a model, we can feed less training data into the model and keep the rest for testing purposes. A low error margin on training data with high error on test data is a clear indication of an overfit model.



Figure 5.2: Overfit

5.2.3 Optimum Model

The optimum model tends to keep both train and test error relatively close by or completely overlapping ensuring proper validation on any data that the model has not seen before.

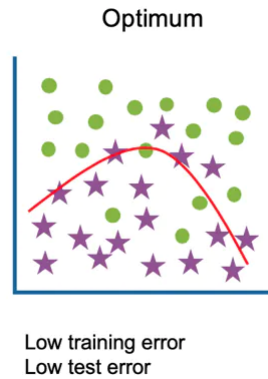


Figure 5.3: Optimum

5.3 Validation Loss & Accuracy Comparison With Different Learning Rate

As previously mentioned, we implemented quite a few deep learning models to perform on our dataset. Each model provided us with different results on the same dataset. A detailed comparison of Validation Loss & Accuracy across the implemented models are given below for reference.

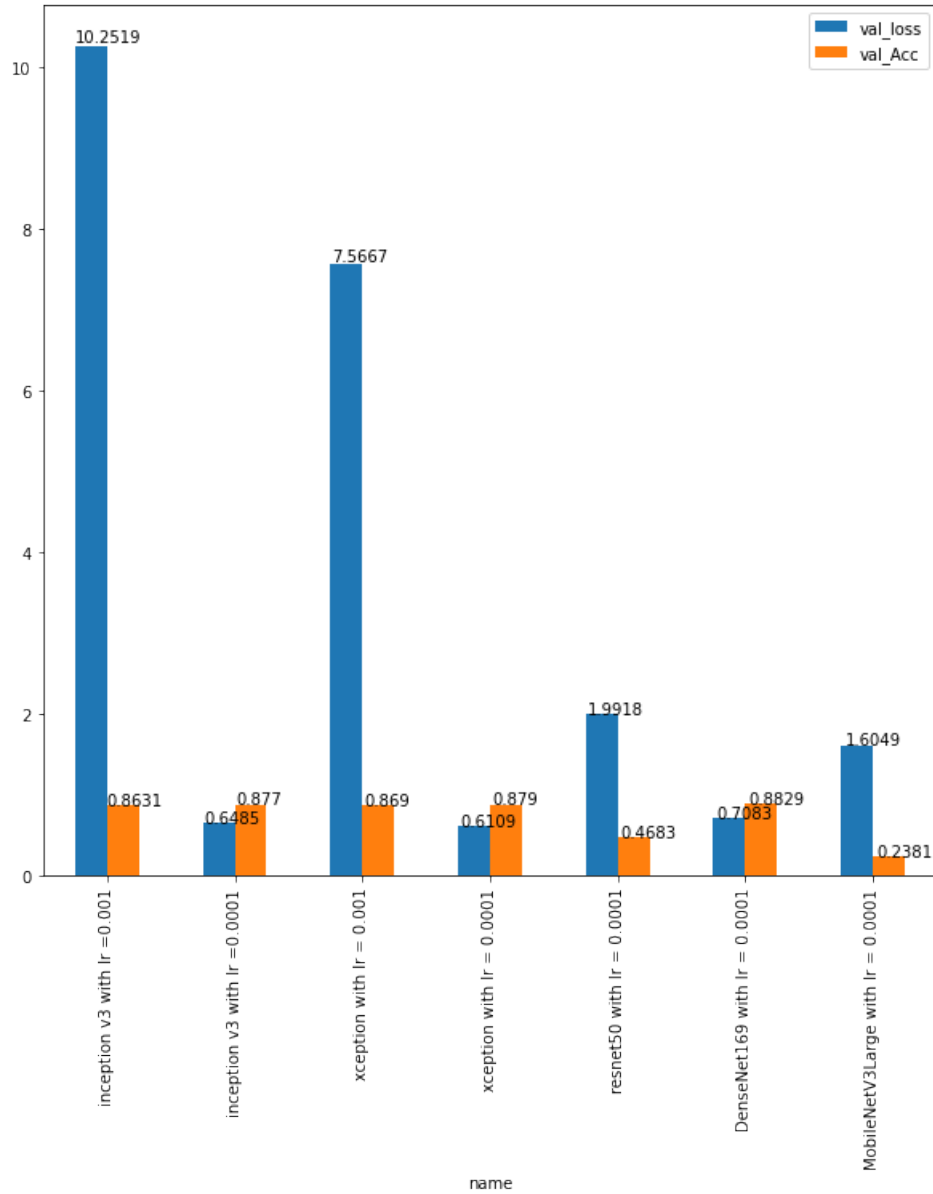


Figure 5.4: Validation Loss & Accuracy Comparison

From the bar comparison we can clearly notice the high validation loss incurred by different models. This graph clearly indicates that our model is underfit. Referring back to section 4 where we described the features from the image data that are used to identify and classify diabetic retinopathy are very precise and tiny. Such features are hard to pinpoint even for trained professionals. When the same comes to a machine learning model, even more difficulties arise. Too much data might result in overfitting where the model will learn bad parts or noises of the image resulting in incorrect prediction. In our case, we lacked a proper and solid dataset with enough samples and hence, the underfitting issue.

5.3.1 Accuracy Comparison

The bar-chart given below only the accuracy of our proposed models.

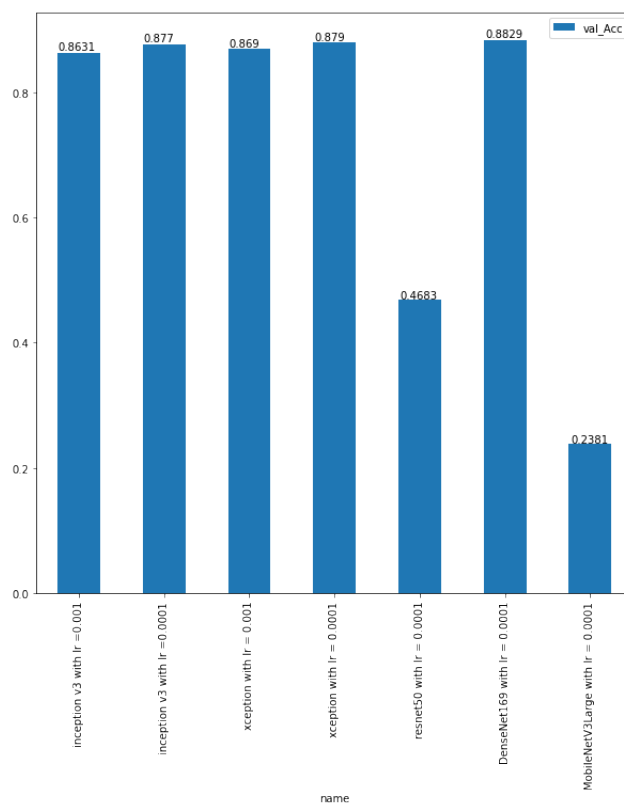


Figure 5.5: Accuracy Comparison

We witnessed the lowest accuracy with MobileNetV3Large while the DenseNet-169 provided us with the highest accuracy.

5.3.2 Loss Comparison

The bar-chart given below only the accuracy of our proposed models. The labels along X-axis denotes different models used with their respective learning rate denoted by lr . From the chart, again, InceptionV3 gives us the highest amount of loss with underfitting. Xception model outputs the lowest amount of validation loss however, in comparison to validation accuracy DenseNet-169 performs way better.

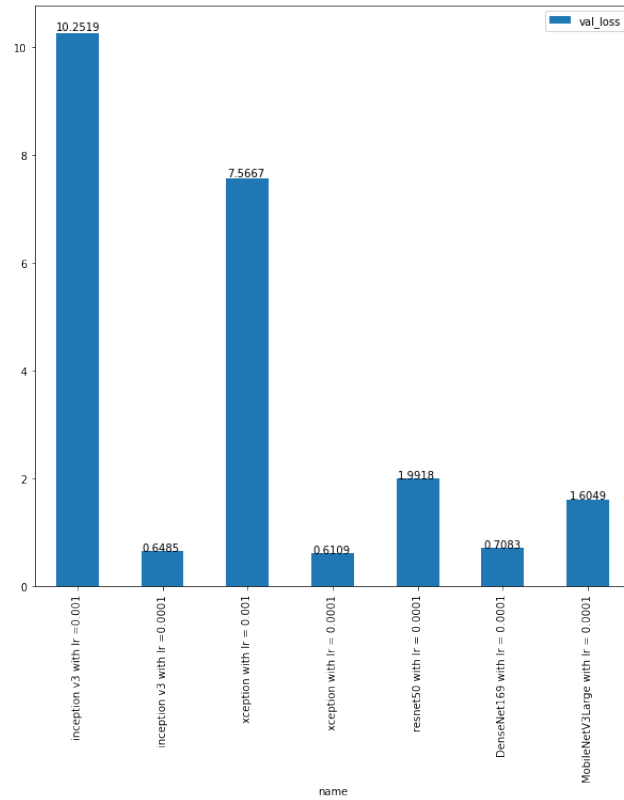


Figure 5.6: Validation Loss

5.4 InceptionV3 With Default Learning Rate

Initially we ran InceptionV3 on our dataset without altering the learning parameters. This resulted in heavy underfitting as seen in the loss-accuracy graph given below.

```

Classification Report
      precision    recall  f1-score   support

     0       0.87     0.86     0.87       101
     1       0.84     0.86     0.85       100
     2       0.84     0.81     0.82       100
     3       0.82     0.88     0.85       104
     4       0.87     0.83     0.85        99

 accuracy          0.85         504
 macro avg         0.85         0.85         0.85         504
 weighted avg         0.85         0.85         0.85         504

```

Figure 5.7: InceptionV3 with Learning Rate 0.001

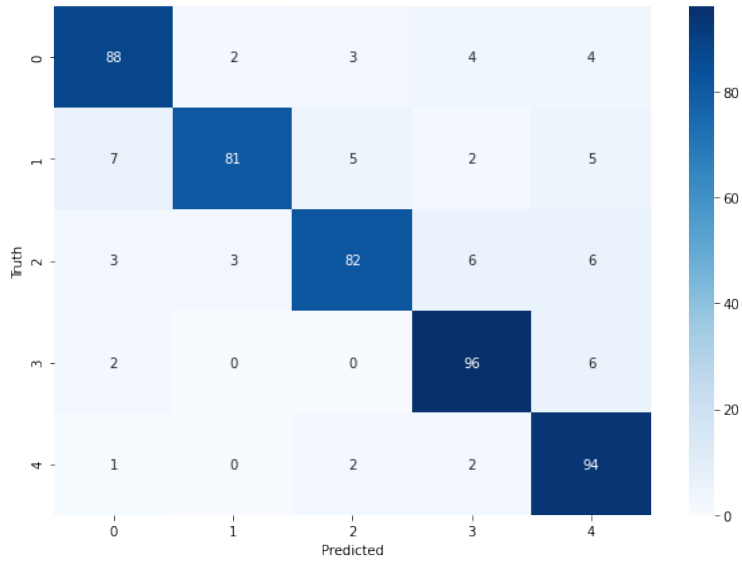


Figure 5.8: InceptionV3 Confusion Matrix Default Learning Rate

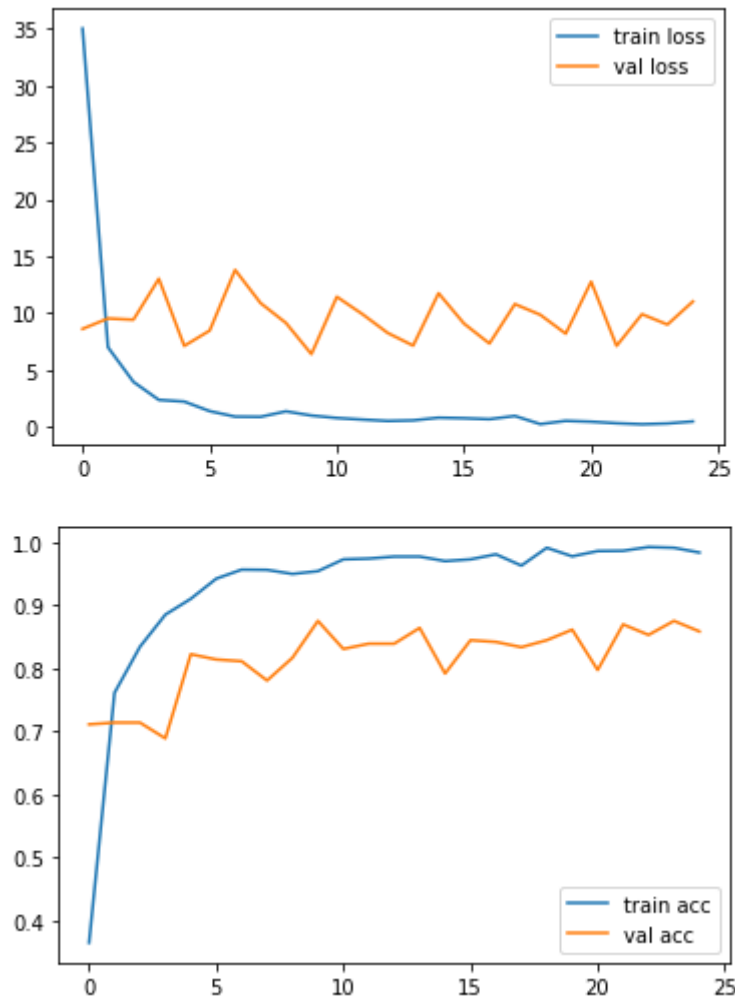


Figure 5.9: InceptionV3 Loss-Accuracy Curve (Default)

5.5 InceptionV3 With Lowered Learning Rate

To avoid the underfitting observed with default learning rate on InceptionV3, we applied Hyper-Parameter Tuning and lowered the learning rate to 0.0001 from 0.001. This change generated significant changes in the outcome of the model.

Classification Report					
	precision	recall	f1-score	support	
0	0.89	0.86	0.87	101	
1	0.90	0.81	0.85	100	
2	0.92	0.80	0.86	100	
3	0.84	0.91	0.88	104	
4	0.80	0.94	0.87	99	
accuracy			0.87	504	
macro avg	0.87	0.86	0.86	504	
weighted avg	0.87	0.87	0.86	504	

Figure 5.10: InceptionV3 with Learning Rate 0.0001

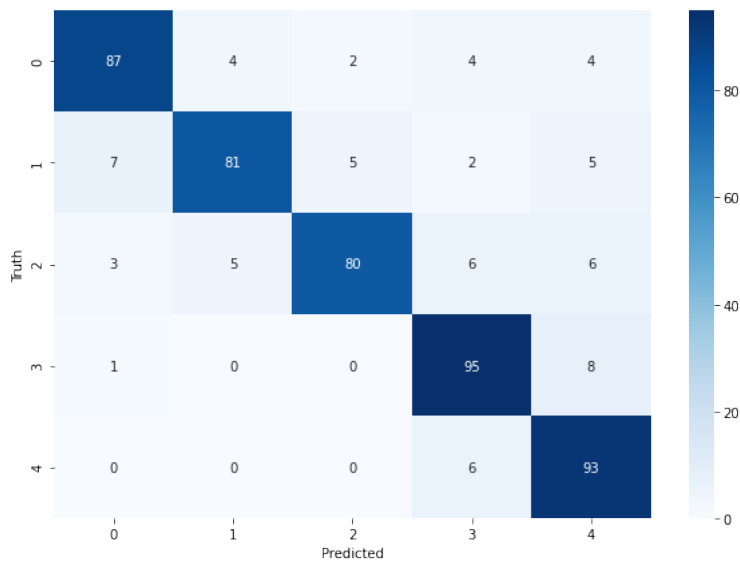


Figure 5.11: InceptionV3 Confusion Matrix Lowered Learning Rate

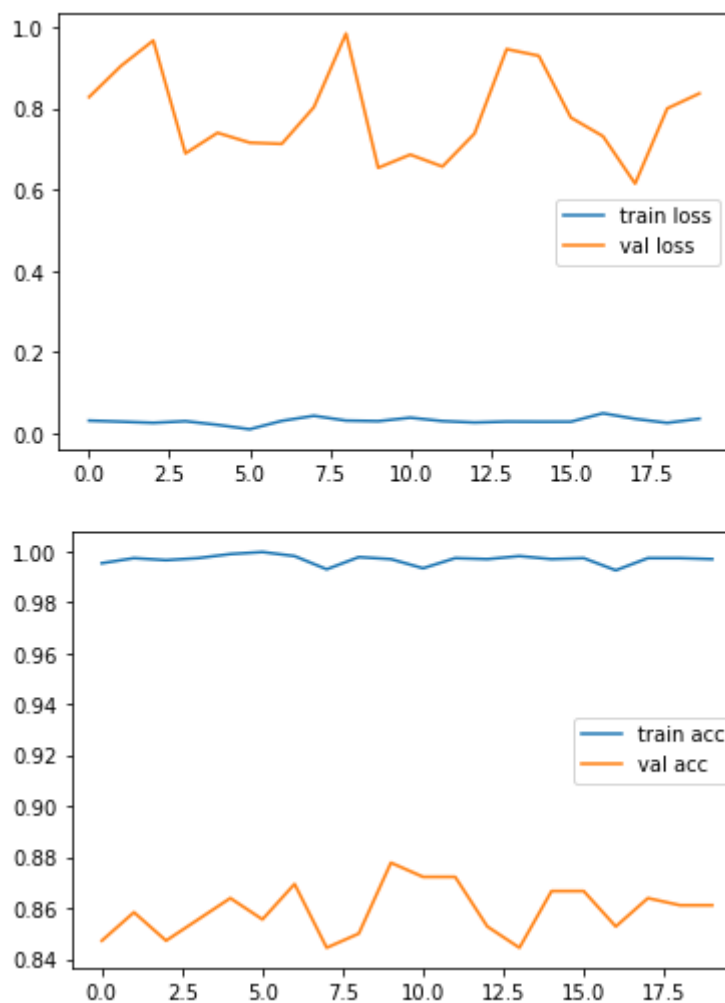


Figure 5.12: InceptionV3 Loss-Accuracy Curve (Default)

5.6 Xception With Default Learning Rate

Similar to the InceptionV3 model, training and testing Xception with default learning rate of 0.001 yielded in high validation loss of 7.57 and a good validation accuracy of 86.9%. This model however faced underfitting.

Classification Report					
	precision	recall	f1-score	support	
0	0.90	0.85	0.87	101	
1	0.89	0.81	0.85	100	
2	0.86	0.80	0.83	100	
3	0.80	0.90	0.85	104	
4	0.84	0.91	0.87	99	
accuracy			0.86	504	
macro avg	0.86	0.85	0.85	504	
weighted avg	0.86	0.86	0.85	504	

Figure 5.13: Xception with Learning Rate 0.001

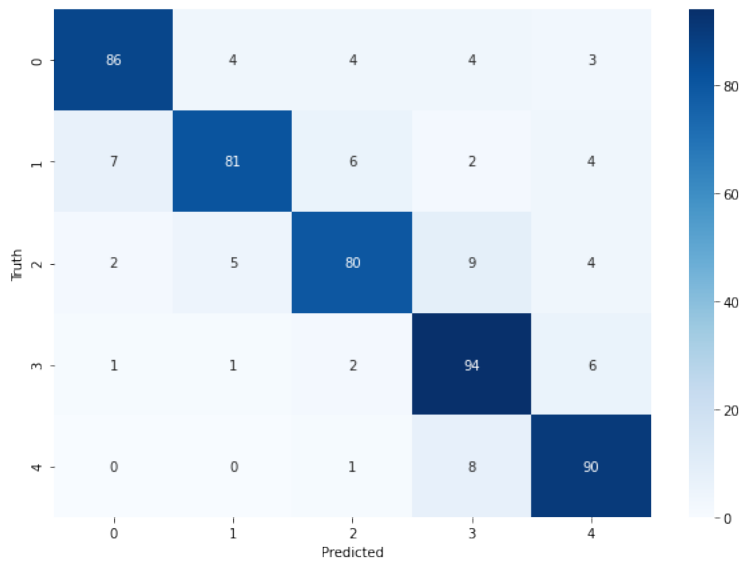


Figure 5.14: InceptionV3 Confusion Matrix Default Learning Rate

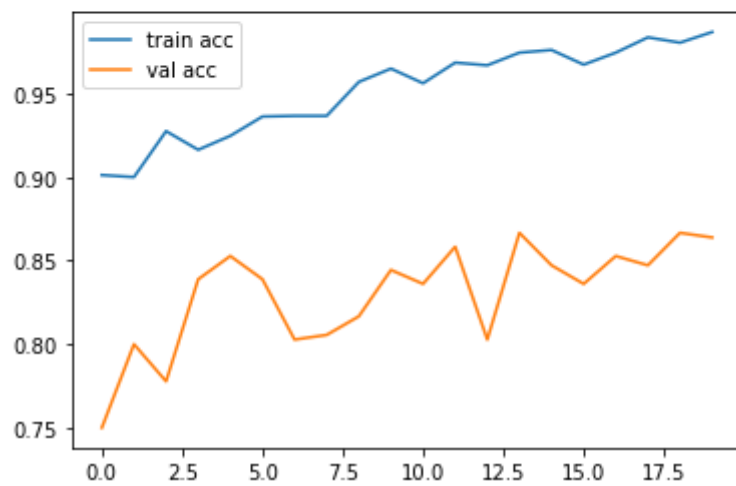
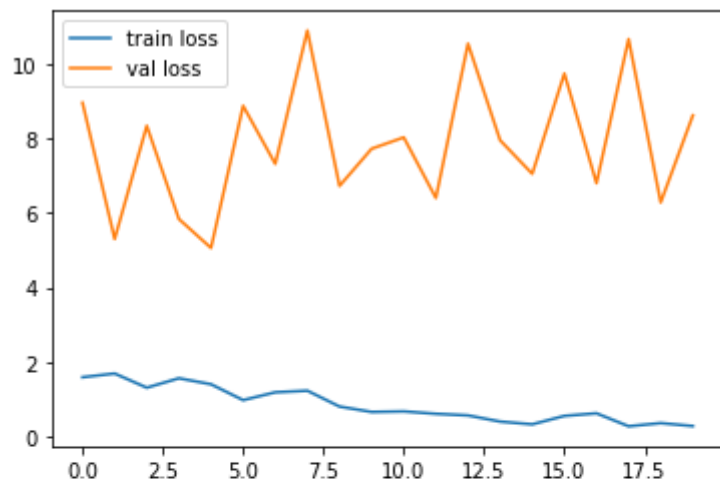


Figure 5.15: Xception Loss-Accuracy Curve (Default)

5.7 Xception With Lowered Learning Rate

To overcome underfitting, lowering the learning rate on Xception to 0.0001 yielded much less validation loss, which is lowest among all 5 models and a even higher validation accuracy than before, 87.9%.

Classification Report					
	precision	recall	f1-score	support	
0	0.95	0.85	0.90	101	
1	0.89	0.84	0.87	100	
2	0.86	0.84	0.85	100	
3	0.81	0.87	0.84	104	
4	0.84	0.93	0.88	99	
accuracy			0.87	504	
macro avg	0.87	0.87	0.87	504	
weighted avg	0.87	0.87	0.87	504	

Figure 5.16: Xception with Learning Rate 0.0001

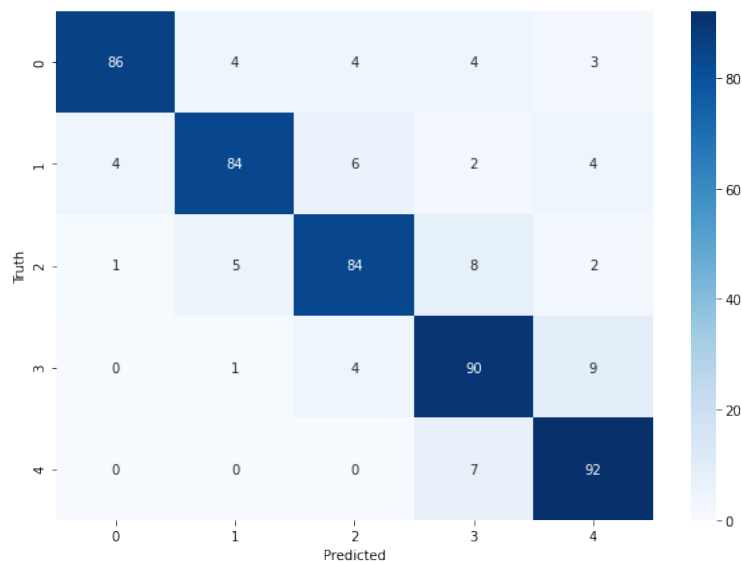


Figure 5.17: InceptionV3 Confusion Matrix Lowered Learning Rate

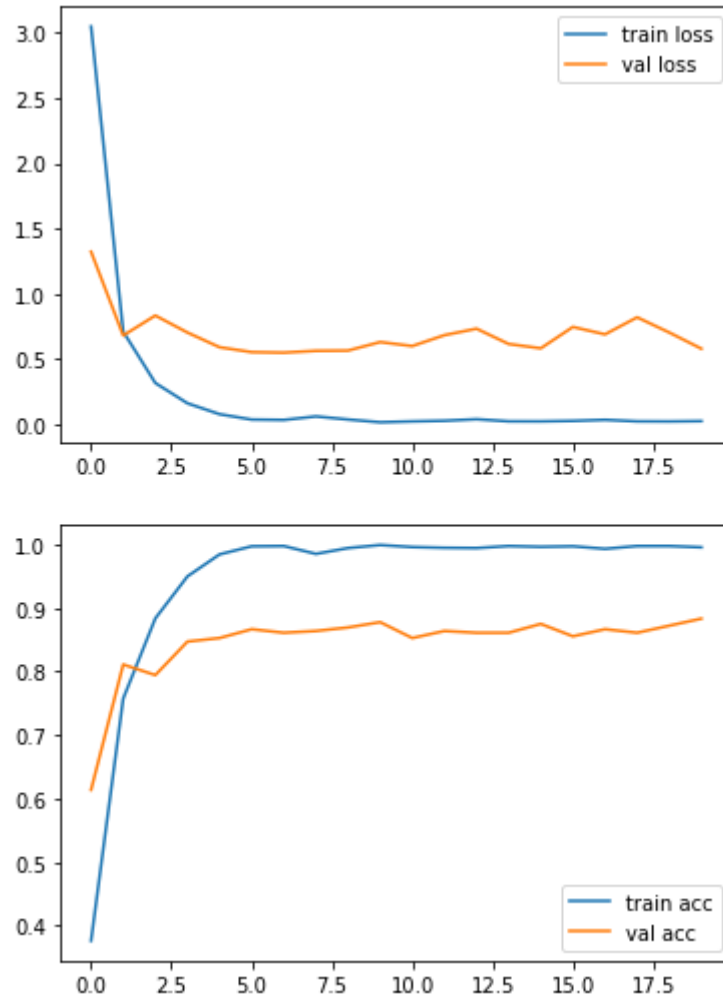


Figure 5.18: Xception Loss-Accuracy Curve (Lowered)

5.8 ResNet With Lowered Learning Rate

From different learning rates used in the previous 4 attempts on two architectures, we came to a conclusion that using a lowered learning rate yielded much better results with the data we are handling. So we trained ResNet-50 with the learning rate 0.0001 and witnessed slightly lower validation loss. However, due to the features being very sophisticated and small to pinpoint, this model failed to attain a decent validation accuracy and is by far the second lowest among other models.

Classification Report					
	precision	recall	f1-score	support	
0	0.46	0.48	0.47	101	
1	0.45	0.49	0.47	100	
2	0.41	0.50	0.45	100	
3	0.47	0.45	0.46	104	
4	0.46	0.31	0.37	99	
accuracy			0.45	504	
macro avg	0.45	0.45	0.44	504	
weighted avg	0.45	0.45	0.44	504	

Figure 5.19: ResNet with Learning Rate 0.0001

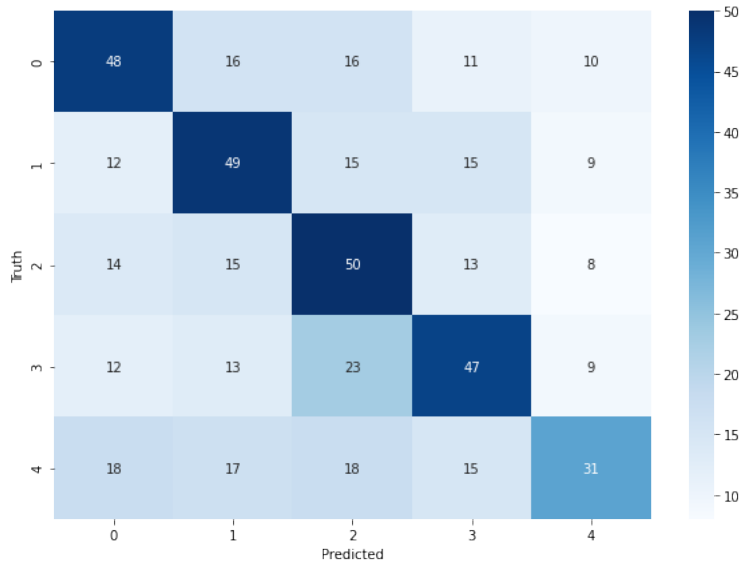


Figure 5.20: ResNet Confusion Matrix Lowered Learning Rate

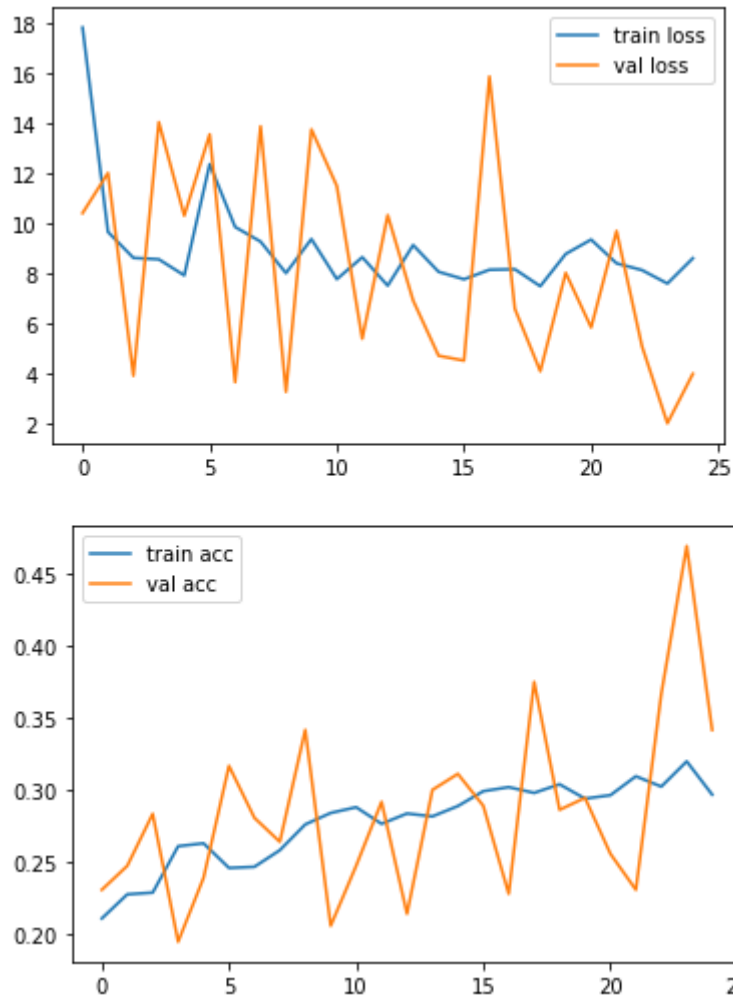


Figure 5.21: ResNet Loss-Accuracy Curve (Lowered)

5.9 DenseNet-169 With Lowered Learning Rate

As previously mentioned, our preferred model the DenseNet-169 outputted the best results in terms of both validation loss and validation accuracy. This architecture gave us the highest accuracy topping at 88.29% while keeping the validation accuracy relatively lower than other models. The learning rate used here was of course, 0.0001.

Classification Report					
	precision	recall	f1-score	support	
0	0.87	0.87	0.87	101	
1	0.94	0.81	0.87	100	
2	0.89	0.82	0.85	100	
3	0.87	0.92	0.90	104	
4	0.82	0.95	0.88	99	
accuracy			0.88	504	
macro avg	0.88	0.87	0.87	504	
weighted avg	0.88	0.88	0.87	504	

Figure 5.22: DenseNet-169 with Learning Rate 0.0001

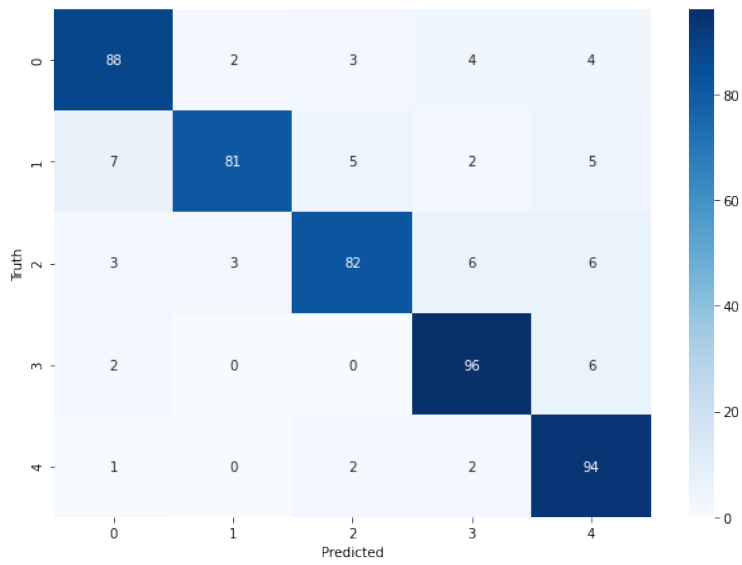


Figure 5.23: DenseNet-169 Confusion Matrix Lowered Learning Rate

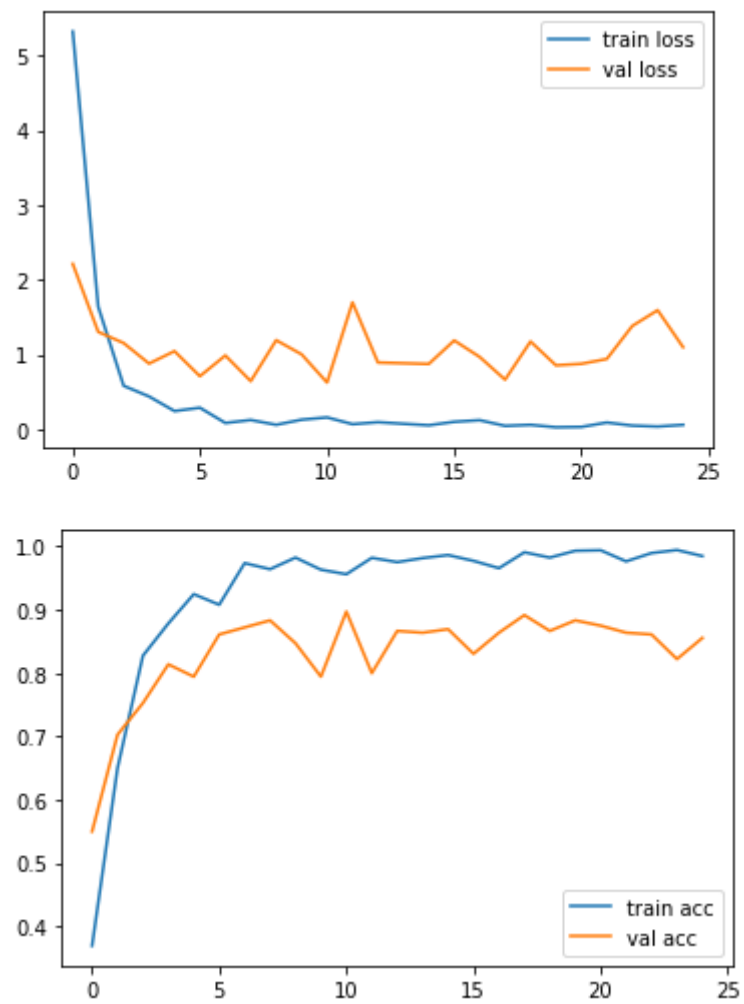


Figure 5.24: DenseNet-169 Loss-Accuracy Curve (Lowered)

5.10 MobileNetV3 Large With Lowered Learning Rate

To further strengthen our claim on DenseNet-169 being the better performing model, we implemented one more architecture, MobileNetV3 Large. This model however portrayed very poor performance. While the validation loss was decently low, validation accuracy of this model is way below par, resting at only 23.81%.

Classification Report				
	precision	recall	f1-score	support
0	0.00	0.00	0.00	101
1	0.20	0.20	0.20	100
2	0.20	0.35	0.26	100
3	0.00	0.00	0.00	104
4	0.20	0.47	0.28	99
accuracy			0.20	504
macro avg	0.12	0.20	0.15	504
weighted avg	0.12	0.20	0.15	504

Figure 5.25: MobileNetV3 Large with Learning Rate 0.0001

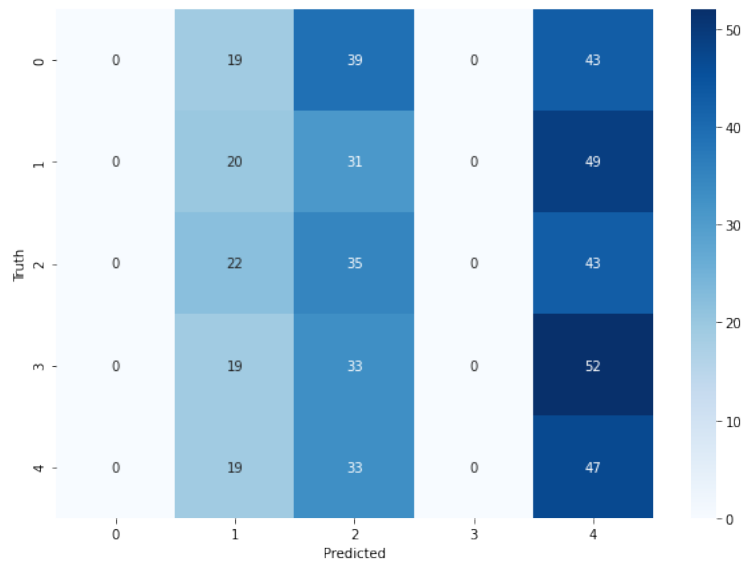


Figure 5.26: MobileNetV3 Large Confusion Matrix Lowered Learning Rate



Figure 5.27: MobileNetV3 Large Loss-Accuracy Curve (Lowered)

5.11 Comparison & Verdict

After considering and comparing all the algorithms used to test and train our dataset, we can come to a conclusion that DenseNet-169 is the better performing algorithm than the rest. DenseNet with a learning rate of 0.0001 results in the closest outcome to an optimum model. Where InceptionV3 and Xception struggles with underfitting, DenseNet excels in all regards. MobileNetV3 Large is in fact a very shallow architecture hence the underfitting as seen with a very high loss and extremely low validation accuracy.

Yashal Shakti Kanungo, et al [9] used InceptionV3 on 10,000 fundus retinal images of Diabetic Retinopathy and attained an accuracy of 67%. On another research conducted by Gabriel Garcia, et al [35] to detect Diabetic Retinopathy with their networks “VGG16”, “VGG16noFC1” and “VGG16noFC2” gained accuracy of 74.3%, 72.70% and 83.68% respectively.

Validation loss of a model is proportional to that models prediction confidence. A model with lower validation loss is more capable of accurately predicting from test

data. DenseNet shows significantly less validation loss and hence it is our proposed model of choice.

Chapter 6

Conclusion

Our model proposes a novel method of Diabetic Retinopathy detection. By using the Adamax Optimizer and Hyper-Parameter Tuning on the deep learning architecture, we were able to achieve a highly accurate prediction rate. While the proposed model with such a small data sample has already shown great results, we believe that with a more balanced dataset, it will be able to perform even better than other proposed models. We believe that our proposed model has high potential and can yield greater accuracy with very little loss. But for that, we require better dataset. Datasets that were available to us to use in this research varied in various ways which resulted in model underfitting and in some cases overfitting. Features of Diabetic Retinopathy are sophisticated, extremely tiny and precise. Without a large amount of data with various lighting conditions to help the model properly predict despite however the test data is. We want to continue this research to fine tune the proposed model a higher degree. The epidemic has prevented us from collecting data, in this instance funduscopy photos from local sources, which we rely on for our research. As a result, we relied on easily available dataset from Kaggle and IDRiD to test and train our model.

Bibliography

- [1] , *Diabetic Retinopathy*, Feb. 2015. [Online]. Available: <https://www.kaggle.com/c/diabetic-retinopathy-detection>.
- [2] P. Porwal, S. Pachade, R. Kamble, *et al.*, *Indian Diabetic Retinopathy Image Dataset (IDRiD)*, Nov. 2019. [Online]. Available: <https://ieee-dataport.org/open-access/indian-diabetic-retinopathy-image-dataset-idrid>.
- [3] Saito and Nakano, “Medical diagnostic expert system based on PDP model,” *IEEE International Conference on Neural Networks*, 1988. DOI: 10.1109/icnn.1988.23855.
- [4] S. Abu-Naser and A. O., “An expert system for diagnosing eye diseases using clips,” *Journal of Theoretical and Applied Information Technology*, vol. 4, Jan. 2008.
- [5] P. Treigys and V. Šaltenis, “Neural network as an ophthalmologic disease classifier,” *Information technology and control*, vol. 36, Jan. 2007.
- [6] J. C. Hwang, A. C. Yu, D. S. Casper, J. Starren, J. J. Cimino, and M. F. Chiang, “Representation of Ophthalmology Concepts by Electronic Systems,” *Ophthalmology*, vol. 113, no. 4, pp. 511–519, 2006. DOI: 10.1016/j.ophtha.2006.01.017.
- [7] M. Usman Akram, S. Khalid, A. Tariq, S. A. Khan, and F. Azam, “Detection and classification of retinal lesions for grading of diabetic retinopathy,” *Computers in Biology and Medicine*, vol. 45, pp. 161–171, 2014. DOI: 10.1016/j.compbimed.2013.11.014.
- [8] B. D. Sullivan, L. A. Crews, E. M. Messmer, *et al.*, “Correlations between commonly used objective signs and symptoms for the diagnosis of dry eye disease: clinical implications,” *Acta Ophthalmologica*, vol. 92, no. 2, pp. 161–166, 2012. DOI: 10.1111/aos.12012.
- [9] Y. S. Kanungo, B. Srinivasan, and S. Choudhary, “Detecting diabetic retinopathy using deep learning,” *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 2017. DOI: 10.1109/rteict.2017.8256708.
- [10] A. Osareh, “Automated identification of diabetic retinal exudates in digital colour images,” *British Journal of Ophthalmology*, vol. 87, no. 10, pp. 1220–1223, 2003. DOI: 10.1136/bjo.87.10.1220.

- [11] A. Vartak, S. Kataria, Z. Vala, and D. S. Borde, “Detection of Diabetic Retinopathy using Deep Learning,” en-US, *International Journal of Engineering Research & Technology*, vol. 10, no. 5, Jun. 2021, ISSN: 2278-0181. [Online]. Available: <https://www.ijert.org/research/detection-of-diabetic-retinopathy-using-deep-learning-IJERTV10IS050333.pdf>,%20https://www.ijert.org/detection-of-diabetic-retinopathy-using-deep-learning (visited on 01/10/2022).
- [12] A. Ortiz, J. Munilla, J. M. Górriz, and J. Ramírez, “Ensembles of Deep Learning Architectures for the Early Diagnosis of the Alzheimer’s Disease,” *International Journal of Neural Systems*, vol. 26, no. 07, p. 1650025, 2016. DOI: 10.1142/s0129065716500258.
- [13] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [14] F. Grassmann, J. Mengelkamp, C. Brandl, *et al.*, “A Deep Learning Algorithm for Prediction of Age-Related Eye Disease Study Severity Scale for Age-Related Macular Degeneration from Color Fundus Photography,” *Ophthalmology*, vol. 125, no. 9, pp. 1410–1420, 2018. DOI: 10.1016/j.ophtha.2018.02.037.
- [15] K. Prasad, P. S. Sajith, M. Neema, L. Madhu, and P. N. Priya, “Multiple eye disease detection using Deep Neural Network,” *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, 2019. DOI: 10.1109/tencon.2019.8929666.
- [16] D. Doshi, A. Shenoy, D. Sidhpura, and P. Gharpure, “Diabetic retinopathy detection using deep convolutional neural networks,” *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, 2016. DOI: 10.1109/cast.2016.7914977.
- [17] W. L. Alyoubi, W. M. Shalash, and M. F. Abulkhair, “Diabetic retinopathy detection through deep learning techniques: A review,” *Informatics in Medicine Unlocked*, vol. 20, p. 100377, 2020. DOI: 10.1016/j.imu.2020.100377.
- [18] D. S. W. Ting, C. Y.-L. Cheung, G. Lim, *et al.*, “Development and Validation of a Deep Learning System for Diabetic Retinopathy and Related Eye Diseases Using Retinal Images From Multiethnic Populations With Diabetes,” *JAMA*, vol. 318, no. 22, p. 2211, 2017. DOI: 10.1001/jama.2017.18152.
- [19] T. Nazir, A. Irtaza, A. Javed, H. Malik, D. Hussain, and R. A. Naqvi, “Retinal Image Analysis for Diabetes-Based Eye Disease Detection Using Deep Learning,” *Applied Sciences*, vol. 10, no. 18, p. 6185, 2020. DOI: 10.3390/app10186185.
- [20] A. W. Setiawan, T. R. Mengko, O. S. Santoso, and A. B. Suksmono, “Color retinal image enhancement using CLAHE,” *International Conference on ICT for Smart Society*, 2013. DOI: 10.1109/ictss.2013.6588092.
- [21] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.
- [22] S. Saha, *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, Dec. 2021. [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

- [23] I. C. Education, *Convolutional Neural Networks*, Jan. 2021. [Online]. Available: <https://www.ibm.com/cloud/learn/convolutional-neural-networks>.
- [24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.
- [25] C. Szegedy, W. Liu, Y. Jia, *et al.*, *Going deeper with convolutions*, 2014. arXiv: 1409.4842 [cs.CV].
- [26] Z. Liu, C. Yang, J. Huang, S. Liu, Y. Zhuo, and X. Lu, “Deep learning framework based on integration of S-Mask R-CNN and Inception-v3 for ultrasound image-aided diagnosis of prostate cancer,” *Future Generation Computer Systems*, vol. 114, pp. 358–367, 2021. DOI: 10.1016/j.future.2020.08.015.
- [27] V. Jain, *Everything you need to know about MobileNetV3 - Towards Data Science*, Dec. 2021. [Online]. Available: <https://towardsdatascience.com/everything-you-need-to-know-about-mobilenetv3-and-its-comparison-with-previous-versions-a5d5e5a6eeaa>.
- [28] Z. Akhtar, *Xception: Deep Learning with Depth-wise Separable Convolutions*, Mar. 2021. [Online]. Available: <https://iq.opengenus.org/xception-model/>.
- [29] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” *CoRR*, vol. abs/1608.06993, 2016. arXiv: 1608.06993. [Online]. Available: <http://arxiv.org/abs/1608.06993>.
- [30] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].
- [31] P. Schratz, J. Muenchow, E. Iturritxa, J. Richter, and A. Brenning, “Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data,” *Ecological Modelling*, vol. 406, pp. 109–120, 2019. DOI: 10.1016/j.ecolmodel.2019.06.002.
- [32] I. Qureshi, J. Ma, and Q. Abbas, “Recent Development on Detection Methods for the Diagnosis of Diabetic Retinopathy,” *Symmetry*, vol. 11, no. 6, p. 749, 2019. DOI: 10.3390/sym11060749.
- [33] *Diabetic Retinopathy: Causes, Symptoms, Treatment*, Oct. 2021. [Online]. Available: <https://www.aao.org/eye-health/diseases/what-is-diabetic-retinopathy>.
- [34] M. Dubow, A. Pinhas, N. Shah, *et al.*, “Classification of Human Retinal Microaneurysms Using Adaptive Optics Scanning Light Ophthalmoscope Fluorescein Angiography,” *Investigative Ophthalmology Visual Science*, vol. 55, no. 3, p. 1299, 2014. DOI: 10.1167/iovs.13-13122.
- [35] G. García, J. Gallardo, A. Mauricio, J. López, and C. Del Carpio, “Detection of Diabetic Retinopathy Based on a Convolutional Neural Network Using Retinal Fundus Images,” *Artificial Neural Networks and Machine Learning – ICANN 2017*, pp. 635–642, 2017. DOI: 10.1007/978-3-319-68612-7\{-}72.