Ophthalmic Disease Recognition using Machine Learning

Submitted in fulfillment of the requirements
of the degree of

Bachelor of Engineering

by

Advait Patil (60001180004)
Jay Shah (60001180021)
Nikhil Soni (60001180035)

Project Guide:
Prof. Sejal Kadam

Electronics Engineering
Dwarkadas J. Sanghvi College of Engineering
Mumbai University
2021-2022

Certificate

This is to certify that the project entitled **“Ophthalmic Disease Recognition using Machine Learning”** is a bonafide work of **Advait Patil(60001180004), Jay Shah(60001180021), Nikhil Soni(60001180035)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor Of Engineering”** in Electronics Engineering.

Internal Guide	Internal Examiner
Prof. Sejal Kadam	

Head of Department	External Examiner
Dr. Prasad S. Joshi	

Principal

Dr. Hari Vasudevan

Project Report Approval for B. E.

This project report entitled ***Ophthalmic Disease Recognition using Machine Learning*** by ***Advait Patil, Jay Shah, Nikhil Soni*** is approved for the degree of **Electronics Engineering**.

Examiners

1.--

2.--

Date:

Place: Vile Parle(west), Mumbai.

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Advait Patil 60001180004

Jay Shah 60001180021

Nikhil Soni 60001180035

Date: 21-04-2022

Abstract

With the popularization of computer-aided diagnosis (CAD) technologies, more and more deep learning methods are developed to facilitate the detection of ophthalmic diseases. Ophthalmic disease detection is a technique used to detect the common eye diseases using fundus images. It is difficult to implement traditional methods which require screening and also availability of human monitoring to all sites. Hence, the need for reliable automatic systems for detecting ophthalmic diseases emerges. The deep learning-based detections for some common eye diseases, including Diabetic Retinopathy, Glaucoma, and age-related macular degeneration (AMD), can be analysed. Generally, the dataset of images obtained are in the RGB form which might not give satisfactory performance. Therefore, it is advised that the dataset needs to be processed from RGB to gray images for better detection of the diseases.

Here, we propose a deep learning model combined with a novel mixture loss function to automatically detect eye diseases, through the analysis of retinal fundus colour images. We are using Convolution Neural networks (CNN) as it is applied to analyse visual imagery and it requires only a little pre-processing. ResNet 50 model is chosen as it overcomes the gradient effect i.e., it uses the skip connection method for some layers whose weight change for the current epoch is significantly less so that the accuracy of the model is not affected. It is used to obtain a greater accuracy as compared to other models.

Acknowledgments

The making of project involves teamwork, vision, patience, and perseverance on the part of group members. But a team can achieve a greater height only by proper guidance from faculty members and friends. Hence, we would like to express our gratitude to all those who in instrumental during the course of developing of the project.

First we would like to thank our internal guide **Prof Sejal Kadam** for her valuable inputs and timely guidance. She always had very practical suggestions and sound knowledge.

Lastly, we would like to thank our head of department, **Dr. Prasad S. Joshi** for extending his support towards the completion of this project and for the timely encouragement.

Advait Patil 60001180004

Jay Shah 60001180021

Nikhil Soni 60001180035

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Ophthalmic Diseases

The project's main goal is to identify or distinguish between three types of ocular disease: Diabetic Retinopathy (DR), Age Related Macular Degeneration (AMD), and Glaucoma. The following is a synopsis of all three diseases:

Diabetic Retinopathy (DR) is the most common cause of vision loss in middle-aged people. The most extensively used technique for preventing blindness is screening for early detection and referral. Due to funding constraints and the lack of human monitoring at all sites, implementing screening programmes as needed remains a challenge. As a result, reliable automatic techniques for detecting DR become necessary.

One of the most common causes of vision impairment in the elderly is age-related macular degeneration (AMD). The choroid and outer layers of the retina, including the retinal pigment epithelium (RPE), are especially vulnerable. Damage to the photoreceptor-containing retinal layers could result in visual loss. The Age-Related Eye Disease Study (AREDS) simplified scale is one of the most regularly used scales for grading AMD. There are four stages of AMD on this scale: none, early, middle, and advanced/late AMD.

Glaucoma affects the optic nerve over time, and it's usually discovered in one of three ways: 1) increased intraocular pressure, 2) detecting the field of abnormal vision, and 3) calculating the cup-to-disc ratio to estimate optic nerve damage (CDR). The optic disc (OD) is the area where the optic nerve exits the eye and is divided into two parts: the optic cup, which is a bright circular area in the centre of the OD, and the neuroretinal rim, which is the peripheral region around the cup. When the optic nerve fibres in the eye are destroyed owing to glaucoma, the optic disc changes optically, resulting in an enlargement of the cup region known as cupping, which is a glaucoma suspect detection indicator.

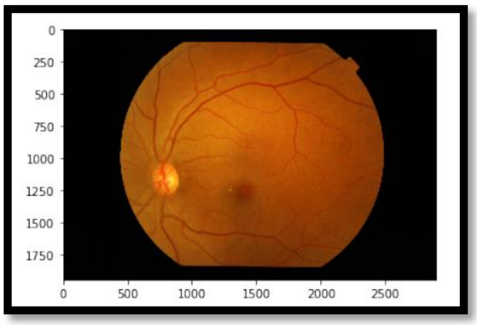


Fig. 1.1: Fundus image of retina

1.2 Aim and Approach

The Image Classification problem is as follows: we are given a series of photos that are all labelled with a single category, and we are asked to forecast these categories for a set of test images and quantify the accuracy of our predictions. This assignment has a number of problems, including viewpoint variation, scale variation, intra-class variance, image deformation, image occlusion, lighting conditions, and backdrop clutter, to name a few.

Most recent publications on ophthalmic data analysis have focused on machine learning and deep convolutional neural networks outperforming legacy algorithms on a variety of applications ranging from the detection and grading of diseases such as AMD, DR, and Glaucoma to the segmentation of anatomical structures such as vessels and retinal layers or intraretinal fluids. We analysed recently published machine learning algorithms in ophthalmology and included datasets that are publically available for research in this subject. The two most commonly discussed issues in these works are classification and segmentation. The classification of eye illnesses leads to more efficient screening programmes, allowing healthcare institutions to reach a greater population.

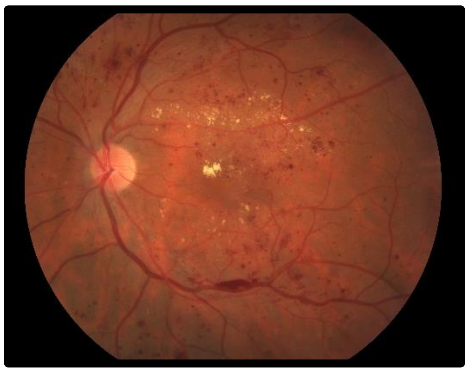


Fig 1.2: Diabetic Retinopathy

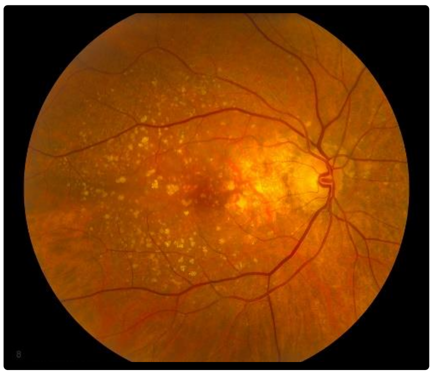


Fig 1.3: Age-related macular degeneration

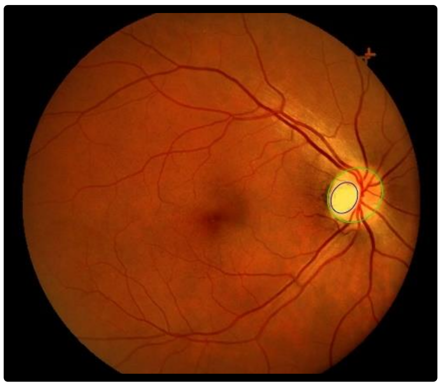


Fig 1.4: Glaucoma

1.3 Convolutional Neural Network

A convolutional neural network (CNN) is an artificial neural network that is specifically intended to analyse pixel data for image recognition and processing. They stand out from other neural networks because they outperform them in terms of picture, audio, and speech signal outputs. CNN's three main layers are:

1. Convolutional layer: The convolutional layer is the block where majority of computation occurs. It is the core building block of the CNN. Some of the important components of CNN are input data, filter, and feature maps. The input data is three dimensional as it is a colour image made up of 3D pixels. So, the RGB in an image will correspond to the height, width and depth of the image. There is also a kernel or a filter which is the feature detector. It is used to check if any specific feature is present or not by moving across various fields of the image. This whole process is known as convolution.
2. Pooling layer: Pooling layers conducts dimensionality reduction, i.e., reducing the number of parameters in input. They are also known as down-sampling. It is like the convolutional layer as it sweeps a filter across the entire input, but there are no weights applied to the filter. Instead of the filter, the output is populated by the kernel which applies an aggregation function within the receptive field.
    a. Max pooling: The filter selects the pixel with the highest value to transmit to the output array as it advances across the input. In comparison to average pooling, this strategy is employed more frequently.
    b. Average pooling: The average value inside the receptive field is calculated as the filter passes over the input and sent to the output array.
3. Fully-connected (FC) layer: In partially connected layers, the pixel values of the input image are not directly connected to the output layer which is possible in the fully connected layers. Based in the features extracted from the previous layers, this layer performs the task of classification. This layer uses softmax activation function to classify the inputs properly. The softmax function produces a range of probabilities which is helpful in determining the weights and priorities to the layers.

Chapter 2

Review of Literature

2.1 Papers Referred

2.1.1 Machine Learning Techniques

M. H. Sarhan et al., "Machine learning techniques for ophthalmic data processing: A review", IEEE J. Biomed. Health Inform., vol. 24, no. 12, pp. 3338-3350, Dec. 2020.

The paper reviews various existing machine learning techniques for ophthalmic data processing. For this, the paper takes into consideration 3 diseases, viz. Diabetic Retinopathy, Age-related Macular Degeneration and Glaucoma. The paper presents a study, wherein the authors have compared different techniques on different datasets.

Table 2.1: Given Datasets and Result

Disease	Task	Results	Dataset
Diabetic Retinopathy	Haemorrhages classification	AUC: 0.97	Kaggle, Messidor-2
Diabetic Retinopathy	Exudates segmentation	Acc: 0.89	DiaretDB0, DiaretDB1
AMD	Classification & segmentation	AUC: 0.99	Moorfield OCT volumes
AMD	Drusen segmentation	IOU: 0.82	Private
Glaucoma	Disk & vessels segmentation	0.95 precision	RIM-ONE
Glaucoma	Retinal vessels segmentation	Acc: 0.94	DRIVE

AUC: Area Under Curve   Acc: Accuracy   IOU: Intersection over union

In the table, you can see that we have listed a few of the models used for classification and segmentation of the parameters used to determine a particular disease. We can see that the accuracy, area under the curve, precision, and intersection of the union of these models are listed and are found to be very high. So, with these model predictions, a classification of the respective diseases is on point.

2.1.2 Retinal Imaging and Image Analysis

Abràmoff, M.D. & Garvin, Mona & Sonka, Milan. (2010). Retinal Imaging and Image Analysis. Biomedical Engineering, IEEE Reviews in. 3. 169 - 208. 10.1109/RBME.2010.2084567.

This paper describes the process of analysing retinal images. It gives a thorough breakdown of retinal images and explains how different kinds of images can help in providing an accurate analysis. It describes the tell-tale signs which help distinguish one condition from the other. It explains in detail 2D and 3D analysis processes of retinal images.

2.1.3 Pulmonary image classification

C. Wang et al.: Pulmonary Image Classification Based on Inception-v3 Transfer Learning Model. IEEE Access, vol. 7, pp. 146533-146541, 2019,
doi: 10.1109/ACCESS.2019.2946000.

This paper proposed a method of lung image classification based on the inception-v3 model-based transfer learning in CT images. It gives an in detailed explanation of the architecture and working of the inception-v3 model. It concludes that the neural network model based on transfer learning performs better in pulmonary image classification. It further demonstrates how a fine-tuning inception-v3 transfer learning can effectively improve the accuracy and how an inappropriate network selection may cause the problem of negative transfer leading to a low accuracy score and the increase of training time. It states that this inception-v3 is best suited for models having high sensitivity and specificity.

2.1.4 Dermoscopy image classification

Zhao, Chen & Shuai, Renjun & Ma, Li & Liu, Wenjia & Hu, Die & Wu, Menglin: Dermoscopy Image Classification Based on StyleGAN and DenseNet201. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3049600

This paper demonstrates on how a DenseNet201 model can be implemented for the classification of lesion images using a GAN-based data augmentation method. It states that the DenseNet201 model was constructed based on the transfer learning models and that the model performs well in the classification task. However, the accuracy can be further improved by using SLA-StyleGAN. The paper provides a good detailed explanation of the DenseNet201 architecture and gives in depth explanation of the working of the model.

2.2 DeepMind Ophthalmology

The Artificial Intelligence (AI) company DeepMind has created an AI model that can predict the absence or presence of a disease known as the Age-related macular degeneration (AMD). The company states that 25% of the people aged more than 60 have dry AMD and that 15% of the people with dry AMD go on to develop exudative AMD. Exudative AMD is a type of AMD which can cause permanent loss of sight. This model can predict the severity of the disease based on the OCT scan of the patient. The eye-scans taken as input are high dimensional scans with 58 million pixels per scan. This input was segmented into thirteen anatomical categories. The segmented data is then combined with the raw data and given as input to the model which can predict the conversion of AMD into exudative AMD within the time stamp of six months. This model uses 2 stage deep neural network models to predict the severity of the disease. By using two stage network, we can get different angle of eye scans for detection. Segmentation of images helps the model to learn some of the anatomical indicators such as drusen (small fatty deposits), or loss of the retinal pigment epithelium. At the end, the model combines the information from the scans and predicts if the eye will progress into ex-AMD or not.

To check if the model -predictions are accurate and up to the clinical standards, they conducted a study with six retinal experts – 3 ophthalmologists and 3 optometrists. As the prediction of ex-AMD is not a routine task performed daily, there was a substantial variability in the assessments of the retinal experts. The model predictions were in some cases better than the predictions of the retinal experts in ex-AMD progression. Due to segmentation, the scan extracts anatomical and pathological features and then provides a systematic method to visualize the change in these tissue overtime. The risk scores provided by the model are accurate enough to give a better picture of ex-AMD conversion.

Chapter 3

Architecture

3.1 Introduction

Convolutional neural networks are a sub-category of neural networks so they have all the features of neural networks. However, CNN is specifically designed to process input images. Their architecture is also more directed: it has two main blocks.

The first block is a very important and characteristic block as it is built to carry out feature extraction. To do this, it performs template matching by applying convolution filtering operations. The first layer filters the image with several convolution kernels and returns “feature maps”, which are then normalized (with an activation function) and/or resized.

This can be done multiple times, until we get the feature maps we desire. The values of the final feature maps are concatenated into a vector which is the output of the first block. It also serves as an input to the second.

The second block is not specific to a CNN. It is a part of all the neural networks used for classification. The input vector values are transformed (with several linear combinations and activation functions) to return a new vector to the output. This last vector contains as many elements as there are classes: element i represents the probability that the image belongs to class i. Each element is therefore between 0 and 1, and the sum of all is worth 1. These probabilities are calculated by the last layer of this block (and therefore of the network), which uses a logistic function (binary classification) or a softmax function (multi-class classification) as an activation function.

The parameters of the layers are determined by gradient backpropagation: the cross-entropy is minimized during the training phase.

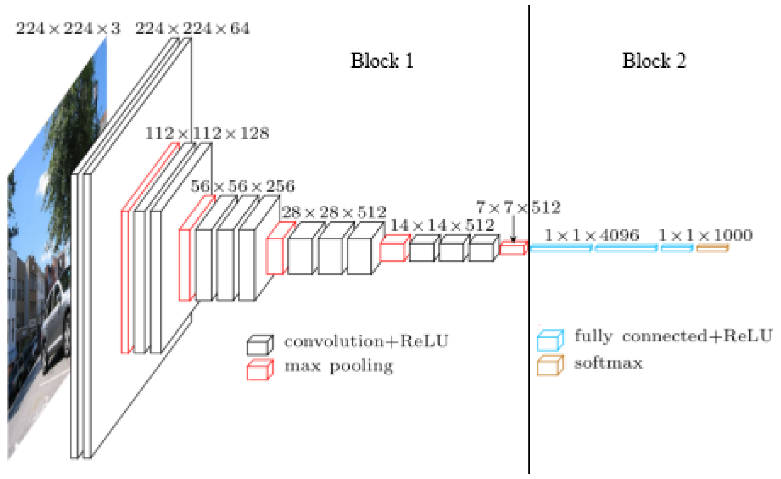


Fig 3.1: Blocks of a CNN

3.2 Different layers of CNN

There are four types of layers for a convolutional neural network: the convolutional layer, the pooling layer, the ReLU correction layer and the fully-connected layer.

3.2.1 Convolutional Layer

The convolutional layer is an important component of CNN and is always the first layer.

The purpose of this layer is to extract relevant features in the input image. This is done by convolution filtering: the principle is to “drag” a window representing the feature on the image, and to calculate the convolution product between the feature and each portion of the scanned image. A feature is then seen as a filter: the two terms are equivalent in this context.

The input to the convolutional layer is that of several images. The convolution of each image with each filter is calculated. The filters match the features which need to be extracted.

For each image-filter pair, we get a feature map, which tells us the presence and location of the features in the image.

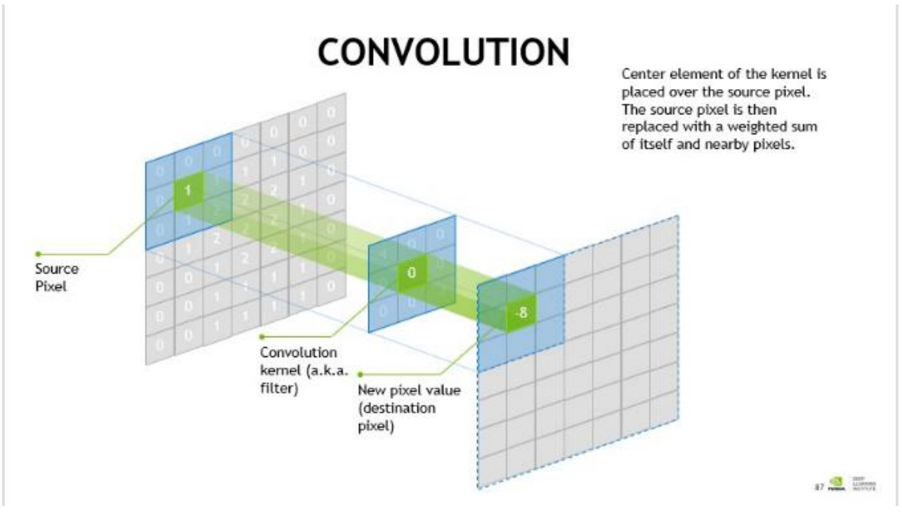

CONVOLUTION
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
Source Pixel
Convolution kernel (a.k.a. filter)
New pixel value (destination pixel)
-8

Fig. 3.2: Convolution

Features are not pre-defined according to a particular formalism (for example SIFT. The training phase of the program helps the algorithm learn about the filters. Filter kernels refer to the convolution layer weights. They are initialized and then updated by backpropagation using gradient descent.

3.2.2 Pooling Layer

This layer receives convolved feature maps and applies pooling to the same. It is placed between two convolution layers. The pooling operation reduces the size of the image while retaining the important features.

This is done by splitting cells into smaller cells. Only the most prominent value from each cell is preserved. If the cell size is too large, then large information is lost. Thus, cells of smaller sizes, like 2x2 or 3x3 are used. They are usually separated by 2 pixels. This prevents loss of information as well as overlapping.

The pooling layer reduces the number of parameters and calculations in the network. This improves the efficiency of the network and avoids over-learning.

3.2.3 ReLu correction layer

ReLU (Rectified Linear Units) refers to the real non-linear function defined by ReLU(x)=max(0,x). Visually, it looks like the following:

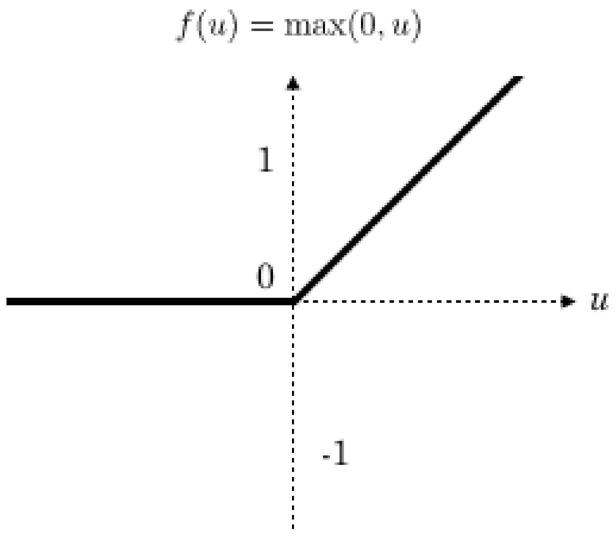


Fig. 3.3: ReLu function

The ReLU correction layer replaces all negative values received as inputs by zeros. It acts as an activation function.

3.2.4 Fully connected layer

The last layer of any neural network is the fully connected layer. This layer produces a new output vector from a given input vector. To do this, it applies a linear combination and then possibly an activation function to the input values received.

The last fully-connected layer classifies the image as an input to the network: it returns a vector of size N, where N is the number of classes in our image classification problem. Each element of the vector indicates the probability for the input image to belong to a class.

The fully connected layer determines the relationship between the position of features in the image and a class. If in an image, a feature which is detected is an indicator of a certain class, then the weight of the corresponding class is given to the image.

3.3 Parameterization of layers

The way the layers are stacked, as well as how they are parameterized, distinguishes one convolutional neural network from another. Convolution and pooling layers do have hyperparameters, or parameters for which the value must first be defined.

The hyperparameters determine the size of the convolution and pooling layers' output feature maps. Each image (or feature map) is $W \times H \times D$, where W is its width in pixels, H is its height in pixels and D the number of channels (1 for a black and white image, 3 for a colour image).

The convolutional layer has four hyperparameters:

1. The number of filters K

2. The size F filters: each filter is of dimensions $F \times F \times D$ pixels.

3. The S step with which you drag the window corresponding to the filter on the image. For example, a step of 1 means moving the window one pixel at a time.

4. The Zero-padding P: add a black contour of P pixels thickness to the input image of the layer. Without this contour, the exit dimensions are smaller. Thus, the more convolutional layers are stacked with $P=0$, the smaller the input image of the network is. We lose a lot of information quickly, which makes the task of extracting features difficult.

For each input image of size W×H×D, the pooling layer returns a matrix of dimensions Wc×Hc×Dc, where:

$$W_c = \frac{W - F + 2P}{S} + 1$$

$$H_c = \frac{H - F + 2P}{S} + 1$$

$$D_c = K$$

Choosing P=F-1/2 and S=1 gives feature maps of the same width and height as those received in the input.

The pooling layer has two hyperparameters:

1. The size F of the cells: the image is divided into square cells of size F×F pixels.

2. The S step: cells are separated from each other by S pixels.

For each input image of size W×H×D, the pooling layer returns a matrix of dimensions Wp×Hp×Dp, where:

$$W_p = \frac{W - F}{S} + 1$$

$$H_p = \frac{H - F}{S} + 1$$

$$D_p = D$$

F=2 and S=2 are good choices for the pooling layer. This removes 75% of the pixels from the input. F=3 and S=2 are also options; in this instance, the cells will overlap. Choosing larger cells resulted in too much information loss and poor results in practise.

3.4 Convolutional neural networks used

The CNN models used in this study are ResNet50, DenseNet201 and InceptionV3. All three of these convolutional neural networks have a different architecture, thus making them work differently and best suited for different kinds of requirements.

3.4.1 ResNet50

Given that Deep Convolutional Neural Networks excel at identifying low, mid, and high level characteristics from images, and that stacking additional layers often improves accuracy, the issue arises: is improving model performance as simple as stacking more layers?

With this question comes the issue of vanishing/exploding gradients, which have been addressed in a variety of ways, allowing networks with tens of layers to converge. However, as deep neural networks begin to converge, we notice another issue: accuracy becoming saturated and then rapidly degrading. This is not caused by overfitting, as one might assume, and adding more layers to a suitable deep model simply increased the training error.

This issue was further resolved by creating a shallower model and a deep model using the shallow model's layers and adding identity layers to it; as a result, the deeper model should not have produced any higher training error than its counterpart because the added layers were only identity layers.

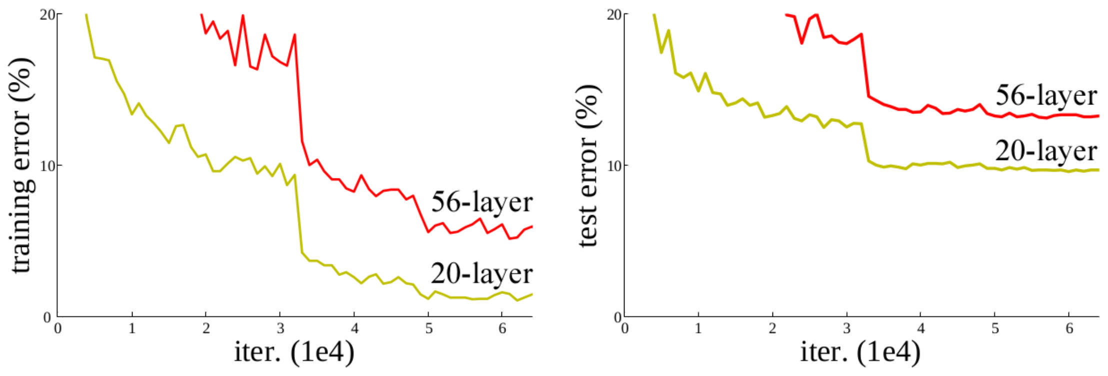


Fig.3.4: Error rates in primitive deep models

On the left and right, we can see that the deeper model is always producing more error, which it shouldn't be doing. This problem was addressed by introducing deep residual learning framework so for this they introduce shortcut connections that simply perform identity mappings.

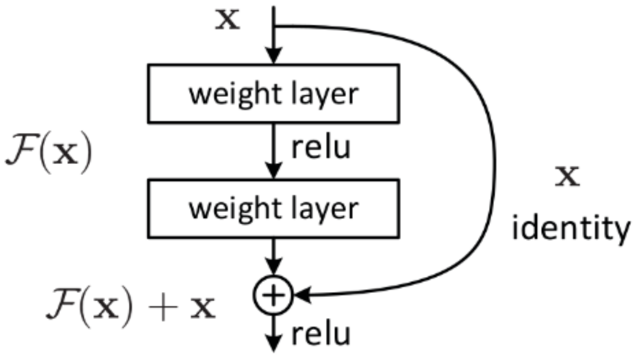


Fig.3.5: Residual mapping

They explicitly let the layers fit a residual mapping and denoated that as H(x) and they let the non linear layers fit another mapping F(x):=H(x)−x so the original mapping becomes H(x):=F(x)+x as can be seen in Figure 3.5.

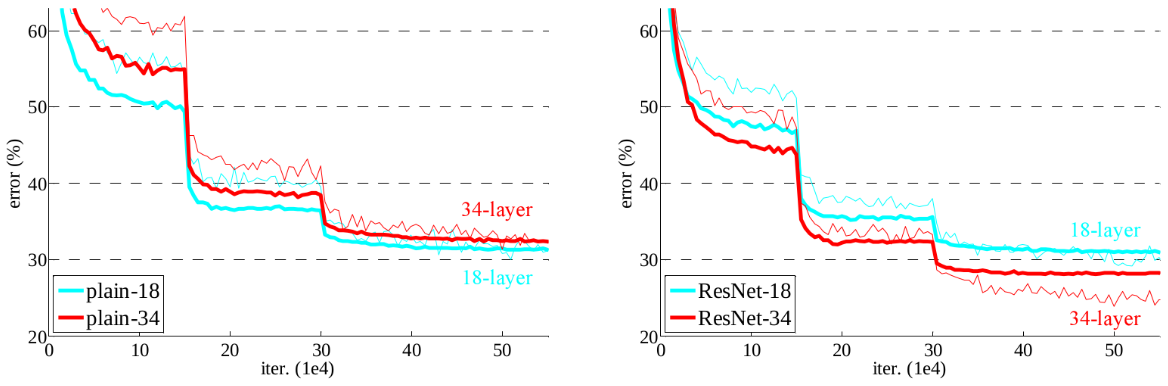
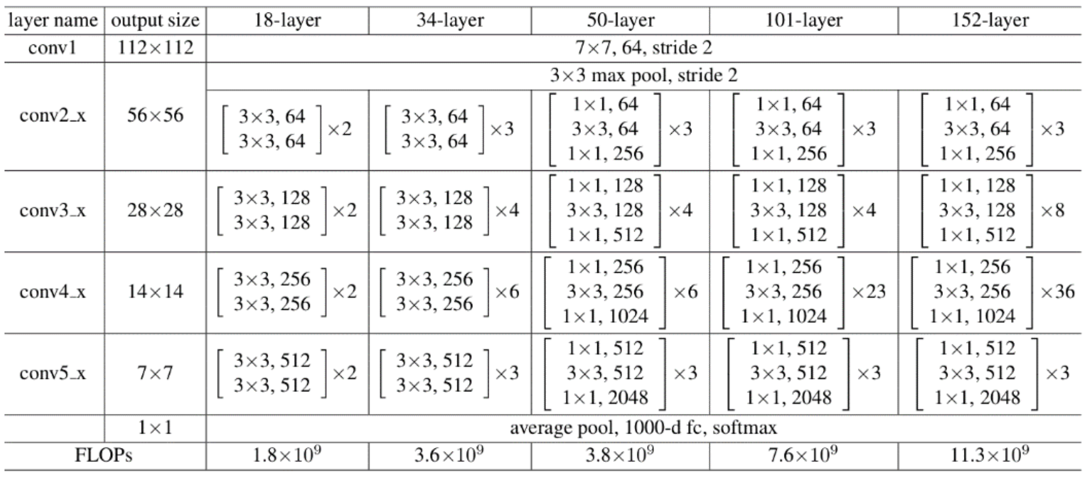


Fig.3.6: Comparison of plain and ResNet models

As can be seen from the given graph, ResNet was able to outperform plan models. Due to a small fundamental tweak which requires no additional components, the efficiency shot up.

For ResNet 50 and higher, a slight change was made: previously, shortcut connections skipped two layers; now, they skip three layers, and 1 * 1 convolution layers have been added.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig.3.7: Architecture of ResNet50

- We have one layer thanks to a convolution with a kernel size of 7 * 7 and 64 distinct kernels, all with a stride of size 2.
- Following that, we have max pooling with a stride size of 2.
- There is a 1 * 1,64 kernel in the next convolution, followed by a 3 * 3,64 kernel, and finally a 1 * 1,256 kernel. These three layers are repeated three times in total, giving us nine layers in this phase.
- Then there's a 1 * 1,128 kernel, followed by a 3 * 3,128 kernel, and finally a 1 * 1,512 kernel. This phase was performed four times, giving us a total of 12 layers.
- Then there's a 1 * 1,256 kernel, followed by 3 * 3,256 and 1 * 1,1024 kernels, which are repeated six times for a total of 18 layers.
- Then a 1 * 1,512 kernel was combined with two more 3 * 3,512 and 1 * 1,2048 kernels, resulting in a total of nine layers.
- After that, we do an average pool and finish with a fully linked layer with 1000 nodes and a softmax function, giving us one layer.
- The activation functions and the max/average pooling layers are not counted.

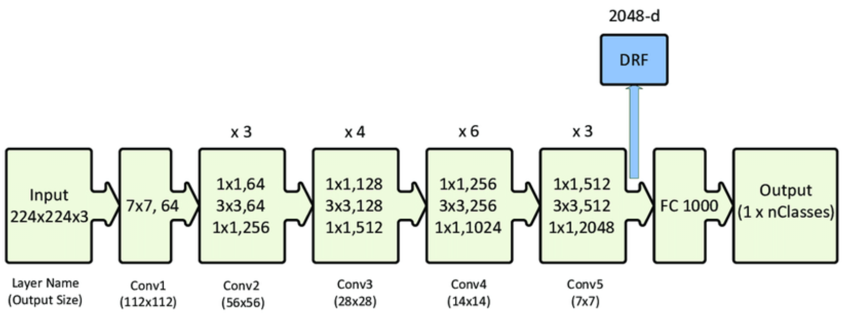


Fig.3.8: ResNet50 block diagram

3.4.2 DenseNet201

Shorter connections between layers close to the input and those close to the output can make convolutional networks significantly deeper, more accurate, and efficient to train. The Dense Convolutional Network (DenseNet) is a feed-forward network that connects each layer to every other layer. Classic convolutional networks with L layers have L connections, whereas traditional convolutional networks with L layers have L connections.

All previous layers' feature-maps are utilised as inputs into each layer, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have a number of compelling advantages, including the elimination of the vanishing-gradient problem, improved feature propagation, feature reuse, and a significant reduction in the number of parameters.

DenseNet has multiple Dense Blocks, which are multiple convolutional layers, all of which are connected to one another. This is the characteristic of DenseNets

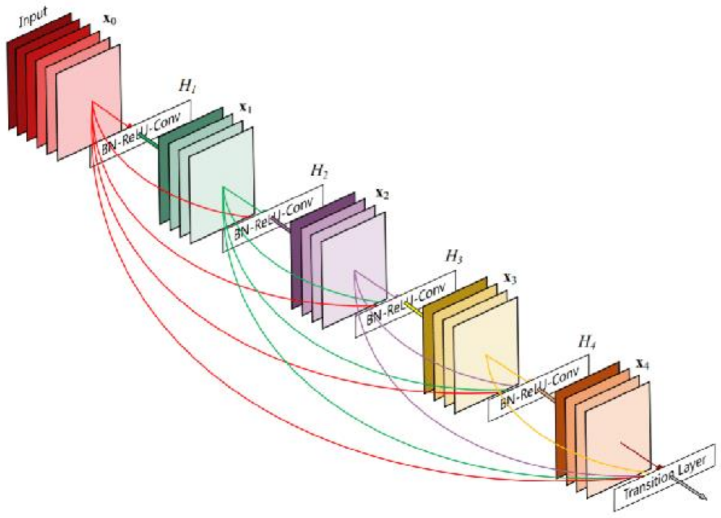


Fig.3.9: A Dense Block consisting of 5 convolutional layers

Adding multiple dense blocks to a network makes the network deeper and more reliable. Feature block sizes in the dense blocks are the same so that they can be concatenated easily. Global average pooling is performed at the end of the last dense block, after which a softmax classifier is attached.

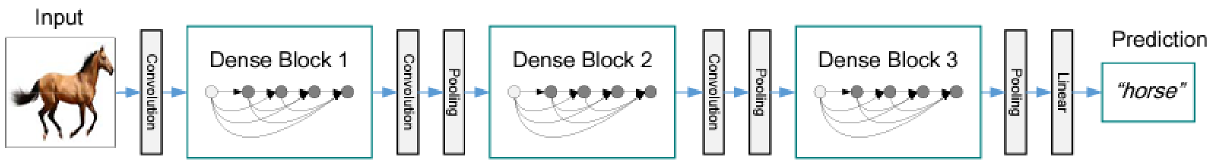
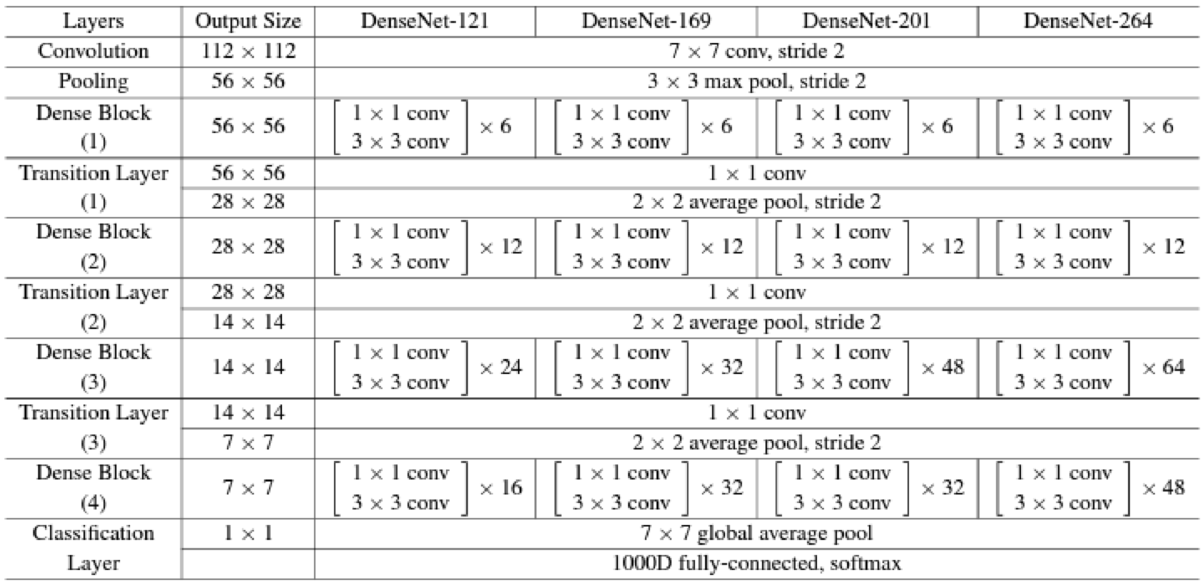


Fig.3.10: A DenseNet with 3 dense blocks



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

Fig.3.11: Architecture of DesneNet

DenseNet has some advantages over other convolutional neural networks:

- Strong gradient flow
- Parameter and computational efficiency
- Diversified features
- Low complexity features

3.4.3 InceptionV3

Inception v3 is a convolutional neural network that was developed as a Googlenet module to aid in picture processing and object detection. It's the third version of Google's Inception Convolutional Neural Network, which was first shown off at the ImageNet Recognition Challenge. Inceptionv3 was created with the goal of allowing deeper networks while keeping the amount of parameters under control: it contains "under 25 million parameters," compared to 60 million for AlexNet.

Inception, like ImageNet, is a library of identified visual objects that aids object classification in the world of computer vision. Many other applications have reused the Inceptionv3 architecture, which is frequently "pre-trained" from ImageNet.

An Inception v3 network's architecture is developed one step at a time, as described below:

1. Factorized Convolutions: This reduces the number of parameters in a network, which improves computational efficiency. It also monitors the network's efficiency.

2. Smaller convolutions: substituting smaller convolutions for larger convolutions results in significantly faster training. A 5 5 filter, for example, has 25 parameters; two 3 3 filters, in place of a 5 5 convolution, have just 18 (3*3 + 3*3) parameters.

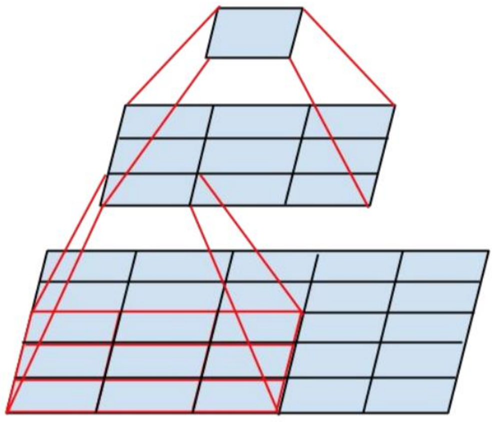


Fig.3.12: Convolutions in InceptionV3

3. Asymmetric convolutions: Instead of a 3 3 convolution, a 1x3 convolution followed by a 3x1 convolution might be used. The number of parameters would be significantly higher than the asymmetric convolution described if a 3x3 convolution was replaced with a 2x2 convolution.

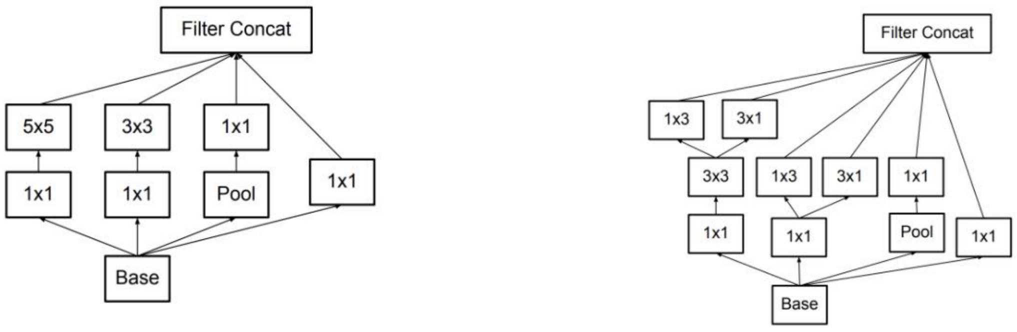


Fig.3.13: Asymmetric convolutions

4. Auxiliary classifier: an auxiliary classifier is a tiny CNN that is introduced between layers during training and adds the loss to the main network loss. Auxiliary classifiers were utilised for a deeper network in GoogLeNet, whereas an auxiliary classifier works as a regularizer in Inception v3.

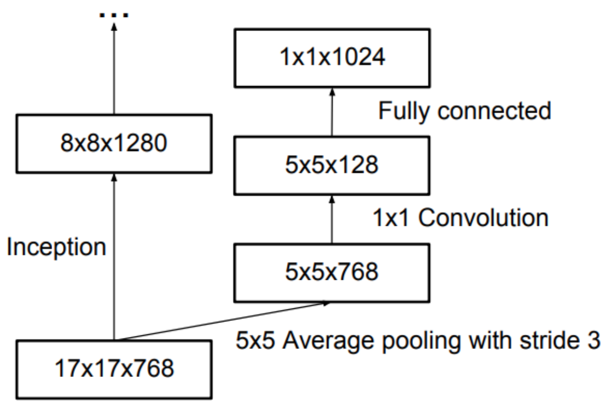


Fig.3.14: Auxiliary classifier

5. Grid size reduction: Pooling techniques are commonly used to reduce grid size. However, a more effective strategy is given to overcome computational cost bottlenecks:

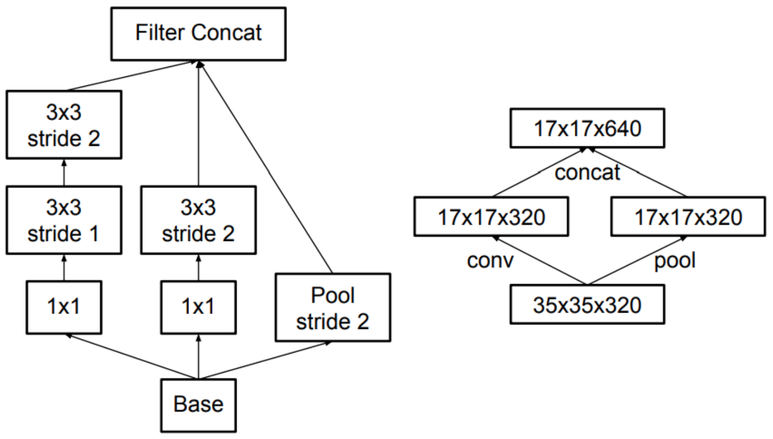


Fig.3.15: Grid size reduction

All the given concepts together make up the final architecture.

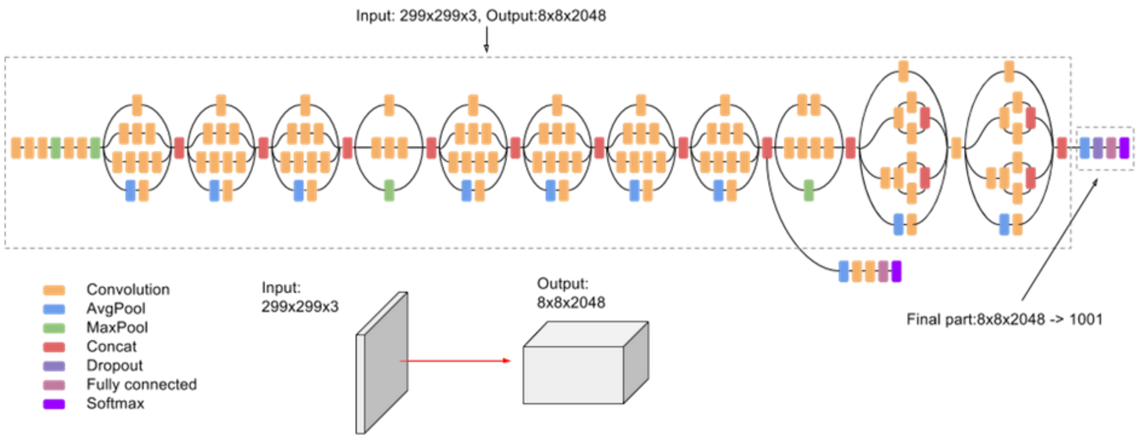


Fig.3.16: Architecture of InceptionV3

Chapter 4

System Implementation

4.1 Data Processing

4.1.1 Data Collection

For the three diseases, the datasets used are – DiaretDB0, ADAM, and REFUGE. The datasets have been mentioned in the referred paper and have thus been chosen. These contain the fundus images of the retina; the following tables describe the contents of the datasets for each of the following diseases.

Table 4.1: Datasets used

Dataset	Disease	No. of images (Healthy)	No. of images (Unhealthy)	Total images
DiaretDB0	Diabetic Retinopathy	22	122	144
ADAM	Age-related macular degeneration	0	126	126
REFUGE	Glaucoma	40	360	400

4.1.2 Grayscale Images

A grayscale image is a single-layered image, as opposed to an RGB image which is composed of three layers. An RGB image is essentially an m*n*3 array whereas a grayscale is an m*n array. Performing operations on a three-layered object is more complex than performing the same operations on a single layer.

It is easier to identify boundaries and specific points/lines in a grayscale image, as compared to an RGB image. This is because boundaries in these images are solely based on luminosity and not color. At the same time, the color in the images is not vital to the identification of the disease, which allows us to remove color as a parameter. Also converting an image from RGB to grayscale reduces the size of the image and hence it becomes easier to store more images in a limited amount of space. The abnormalities can be easily spotted in the grayscale format of the fundus image as compared to the RGB image as can be seen in the example shown below

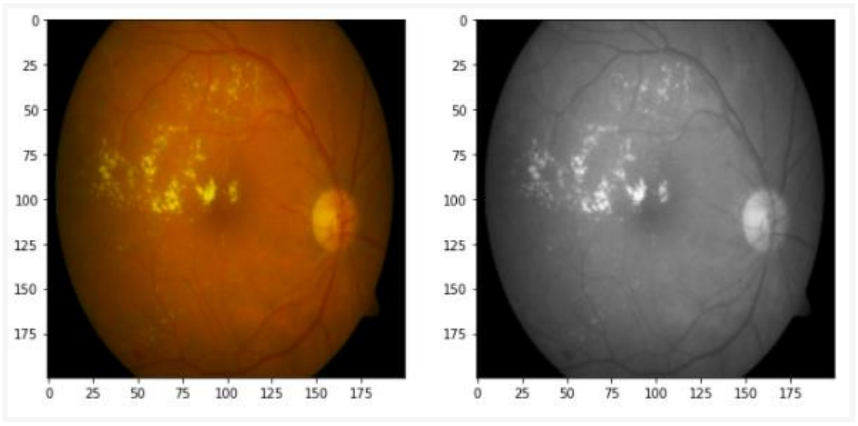


Fig.4.1: Grayscale Image

4.1.3Gaussian Blur

Gaussian blur is a linear filter and is used to remove noise from the image. Its main function is to blur the image, which reduces the noise in it. After applying a gaussian blur to the grayscale image obtained, it blurs out the noise, which in this case is the vitreous gel (fluid in the retina). It highlights the exudates, optical disk, artery, and venules. This image is not clear enough to yield good results, as it is, after all, a blur filter. Thus, the gaussian blur is superimposed on the grayscale image, which gives an image that has the required abnormalities highlighted. As it can be seen, the leftmost image is the original RGB image followed by the grayscale representation of the same image, and lastly on the rightmost side is the final image which is obtained after superimposing the grayscale image on the Gaussian blurred image of the same fundus image. Hence to obtain the highest accuracy and best outcomes, this image will be used to train the models that will detect and classify the three diseases.

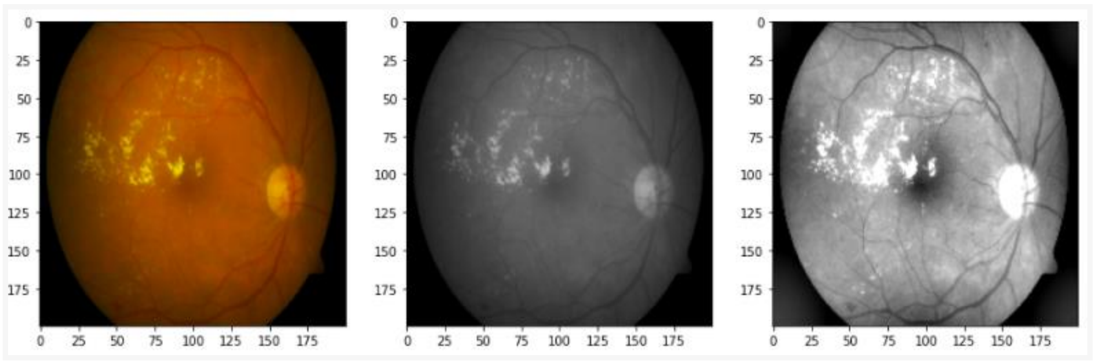


Fig.4.2: Gaussian blurred image superimposed on grayscale image

4.2 Methodology

Before starting with any implementation work, there is a need to make sure that the number of images for each of the class must be equal or close to each other so that class imbalance can be avoided. For this purpose, all the images of the four classes which are Healthy, Diabetic Retinopathy, Age Realted Macular Degeneration and Glaucoma are balanced as close to each other as possible. The below bar graph indicates the same as follows:

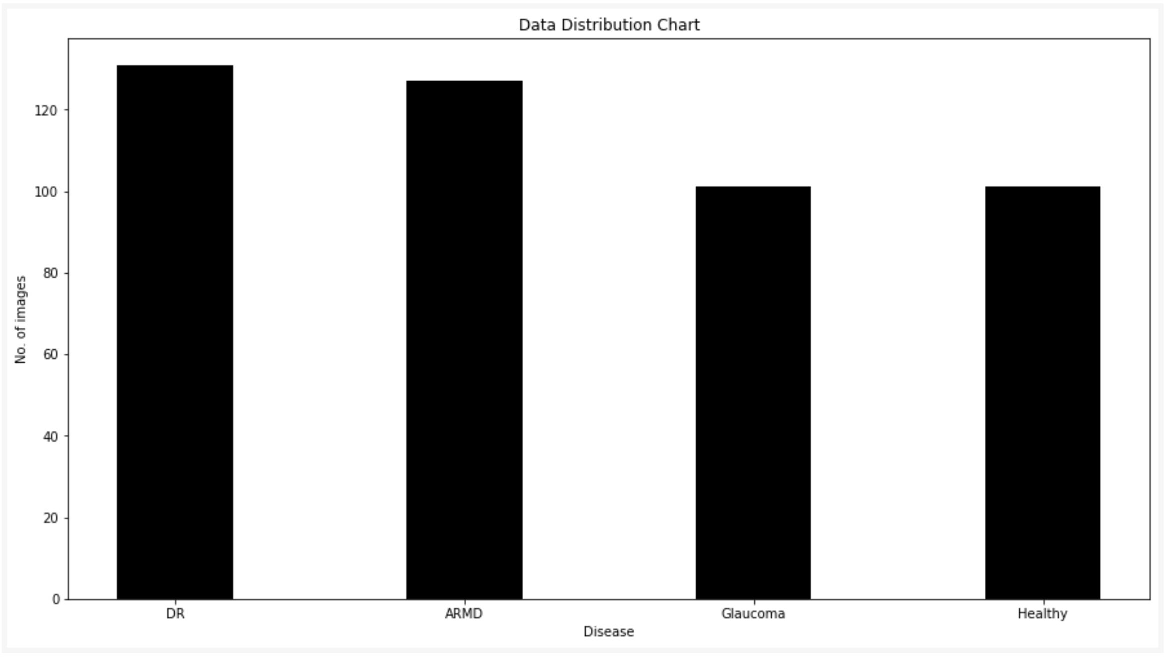


Fig.4.3: Data Distribution Chart for all 4 classes

The implementation starts with Data Preprocessing which is nothing but Image Processing. It is necessary to make sure that there is uniformity in the type of image files so that all of the images can be processed by the same functions. The 5 Types of Digital Image Files are TIFF, JPEG, GIF, PNG, and Raw Image Files. The Dataset collected from all the sources may or may not contain images with the same file format, hence they need to be converted into one form. Therefore, all the images are converted into PNG from whatever form they are in.

As it can be noticed that the main area of concentration in a fundus image is the pixels that are in a circle with the center of the image as the origin. The other side regions of a fundus image do not contribute to determining any evidence of the disease a patient possesses. Due to this, a circular crop function can be applied to get rid of the unnecessary region and concentrate only on pixels that will help in determining and classifying the images in a better way. In the below example, the left-hand side image is the original RGB image and the right-hand side image is the resultant image after applying the circular crop function on the left-hand side image.

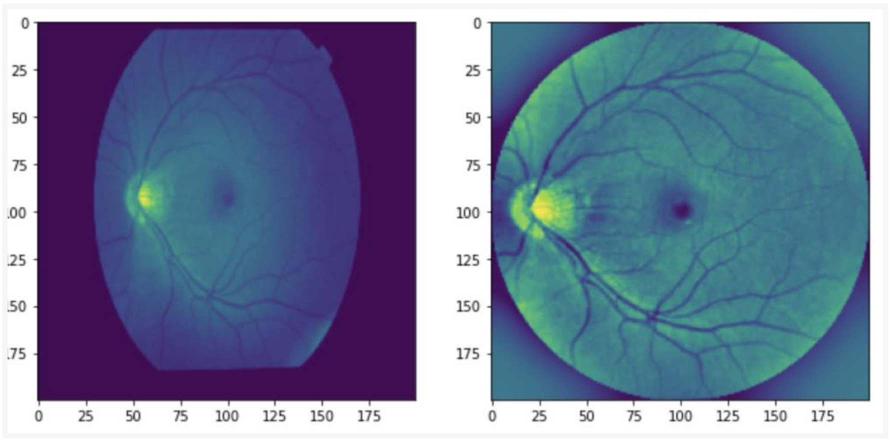


Fig.4.4: Circular crop

Now, the images are ready to be converted into grayscale and then apply the gaussian blur technique to make them ready to be given as input to the model while training and testing.

These preprocessed images are now divided into 3 categories which are Train, Test, and Validation. This splitting of data is done using the split folder library in the following proportion, 60% images as train, 20% images as test, and 20% images as validation. This split is done by randomly selecting images so that all the images are easily shuffled for better evaluation. This completes the preprocessing step and thus the training of the models can be started. But, instead of using the images directly from the directories, Image Data Generators are used because these generators load images in batches which helps in memory usage optimization and better performance.

4.2.1 Image Data Generators

Image Data Generators provide a way to work batch-wise instead of working on the entire dataset at once. The Image Data Generator provides two methods to accept data, they are flow_from_directory and flow_from_dataframe. The former method requires no extra work in creating a data frame and a Data Generator can be directly created using the directory path of our dataset. The later method that is flow_from_dataframe needs a data frame object as an argument to work. Hence, there is a need to create a data frame from the dataset available and then pass it to the method to get an instance of a Data Generator. Due to this reason, in this project, the flow_from_directory method of the Image Data Generator class has been deployed for better convenience. These generators also provide a way to augment our data via different arguments such as rotation_range, brightness_range, zoom_range, horizontal_flip, vertical_flip, samplewise_center, featurewise_center, etc

The syntax to implement Image Data Generator class is as follows:

```
tf.keras.preprocessing.image.ImageDataGenerator(

    featurewise_center=False,

    samplewise_center=False,

    featurewise_std_normalization=False,

    samplewise_std_normalization=False,

    zca_whitening=False,

    zca_epsilon=1e-06,

    rotation_range=0,

    width_shift_range=0.0,

    height_shift_range=0.0,

    brightness_range=None,

    shear_range=0.0,

    zoom_range=0.0,

    channel_shift_range=0.0,

    fill_mode='nearest',

    cval=0.0,

    horizontal_flip=False,

    vertical_flip=False,

    rescale=None,

    preprocessing_function=None,

    data_format=None,
    validation_split=0.0,

    dtype=None

)
```

These are all the available arguments provided to create an instance of the Image_Data_Generator class and use it to train our model. The syntax for flow_from_directory method is:

```
flow_from_directory(

    directory,

    target_size=(256, 256),

    color_mode='rgb',

    classes=None,

    class_mode='categorical',

    batch_size=32,

    shuffle=True,

    seed=None,

    save_to_dir=None,

    save_prefix='',

    save_format='png',

    follow_links=False,

    subset=None,

    interpolation='nearest'

)
```

The directory is the argument where the directory path of the dataset is given to the method. This path can contain PNG, JPG, BMP, PPM or TIF images and all these images will be included in the generator that is been created. The batch size is the size of batches in which image will be processed. The default value for a batch size is 32. Also, the class mode has to be categorical for problems that involve multiple labels.

With the Data Generators in place, the model can now be trained using the training image dataset. Three neural network models from the Keras applications have been compared in this project, they are ResNet50, InceptionV3, and DenseNet201. These are similar models which contain convolutional layers along with some special layers that separate them from one another. The final layer in the model which is the fully connected layer is not included in the base model while defining the model. Instead, other special layers namely GlobalAveragePooling2D and Dense layers have been added at the end of these base models so that they predict as per our problem statement. A brief description about these layers is discussed below.

4.2.2 Global Average Pooling

The global average pooling layer is used to replace the top layer of each model. It is designed to replace the fully connected layers in a CNN network. The global average pooling layer takes the average of each feature map instead of adding fully connected layers on top of the feature map which is then directly fed to the softmax layer. It enforces correspondences between feature maps and categories so that the feature maps can be easily interpreted unlike the fully connected layers. Also, the overfitting of the layer is avoided in this layer as there is no parameter to optimise in the global average pooling. The syntax to implement Global Average Pooling layer is as follows:

```
tf.keras.layers.GlobalAveragePooling2D(

    data_format=None, keepdims=False, **kwargs

)
```

The significance of the parameters stated above are:

- data_format: A string, one of channels_last (default) or channels_first. The ordering of the dimensions in the inputs. channels_last corresponds to inputs with shape (batch, height, width, channels) while channels_first corresponds to inputs with shape (batch, channels, height, width).

- keepdims: A boolean, whether to keep the spatial dimensions or not. If keepdims is False (default), the rank of the tensor is reduced for spatial dimensions. If keepdims is True, the spatial dimensions are retained with length 1.

4.2.3 Dense

This is the most common layer used to build a neural network. The neurons of the dense layer are connected to all the neurons of its preceding layers. The main task of the neurons of dense layer is to perform matrix-vector multiplication. The general rule of matrix-vector multiplication is that the row vector must have as many columns like the column vector. The Dense layer can be easily implemented using keras through the following syntax:

```
tf.keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

The significance of these parameters are as follows:

- Units: The size of the output from the Keras dense layer is defined by this parameter, which is one of the most basic and necessary parameters of the Keras dense layer. It must be a positive integer because it represents the output vector's dimensions.
- Activation: The activation function is a function in neural networks that is used to alter the input values of neurons. It essentially incorporates non-linearity into neural network networks in order for the networks to understand the link between input and output values.
- use_bias: The Use Bias parameter determines whether or not a dense layer should employ a bias vector. It's a boolean argument that defaults to true if it's not defined.
- kernel_initializer: The kernel weights matrix is initialised with this option. The weight matrix is a set of weights multiplied by the input in order to extract relevant feature kernels.
- Bias_initializer: The bias vector is initialised with this parameter. The additional sets of weight that require no input and correlate to the output layer are referred to as bias vectors. It is set to zeros by default.
- Kernel regularizer: If any matrix has been initialised in the kernal initializer, this argument is utilised to regularise the kernel weight matrix.
- Bias_regularizer: If any vector was initialised in the bias initializer, this argument is used to regularise the bias vector. It is set to none by default.
- Activity_regularizer: This option is used to make the activation function that we defined in the activation parameter more regular. It is applied to the layer's output. It is set to none by default.
- kernal_constraint: The constraint function is applied to the kernel weight matrix using this parameter. It is set to none by default.
- Bias_constraint: The constraint function is applied to the bias vector using this parameter. It is set to none by default.

The base model is defined using the concept of Transfer Learning. Neural Network Models can take up to weeks for training on a large dataset. Hence, transfer learning provides a way of reusing already trained models such as ResNet50 and applying them to one's problem statement by making small changes to the original model layers. The ImageNet dataset contains over 1000 classes which means that one can use these imagenet weights for their base model which will help them classify their images as per needs.

4.2.4 ImageNet

To give some priority to the layers, we add weights to the layers. We use ImageNet for that purpose. ImageNet is a standard for image classification. There is a yearly competition with millions of images in thousands of categories and the models are measured against each other for performance. Therefore, a standard measure is provided for how good a model is. For example, if there are no haemorrhages found in the retinal image, it surely cannot be a Diabetic Retinopathy image. So, the weights corresponding to the layers having DR as output would have lesser value than other weights in the same convolution layer. ImageNet decides the weights that we add to the layer while the model decides which layer should be given the priority.

```
weights = 'imagenet'
```

Since multi-class classification has to be done, the SoftMax activation function is used in the output layer of each of the models before training them. This helps in the evaluation and prediction of a multi-nominal probability distribution. Also to avoid the vanishing gradient problem ReLu activation function is used in the hidden layer of the model.

4.2.5 Softmax

The softmax activation function predicts a multinomial probability distribution in the output layer of the network. This function is used for multi-classification problems i.e., for models where there are more than two class labels present. Softmax function is used to normalise the outputs and then convert them into probabilities. Suppose we have a convolution layer whose extreme values are in the range [- 10,10]. Now if we use the softmax activation function, the values are then transformed into a range [0,1] so that they can be interpreted as probabilities and then compared with the expected output where 1 indicates the desired output.

```
predictions = Dense(train_generator.num_classes,
activation='softmax')(x)
```

4.2.6 Rectified Linear Unit (ReLU)

Rectified Linear Unit or the ReLU activation function is a piecewise linear function. The output of this function has only two attributes. If the input to the function is positive, the output will be directly fed to the next layer and if the input is negative, the output of this function will be zero. As it is easier to train and achieves better performance, it is used in many neural network models. There are some exudates or haemorrhages detected in the retinal image, the model concludes that the image is not an image of a healthy person. So, for the layers corresponding to the healthy output the model starts giving out negative values. And the ReLU function then makes it zero which states that the image cannot be a healthy image. This process is done to make the convolution process easier and to train the model faster and achieve greater accuracy and efficiency.

```
x = Dense(1024,activation = 'relu')(x)
```

4.2.7 Adam Optimizer

Adam optimization is based on the adaptive estimation of first and second order moments. Adaptive moment estimation is an optimization technique algorithm for stochastic gradient descent. This method is efficient and requires less memory. So, this method can be used when there are problems involving a lot of data and parameters. This method is a combination of 'RMSP' algorithm and 'Gradient descent with momentum' algorithm. Here, we can control the rate of gradient descent such that when the algorithm reaches the gradient descent, it has minimum oscillations. Also, we need to take big enough steps such that the local minima would not be an issue. By combining both the above-mentioned algorithms, we can reach the global minimum efficiently with low cost and better performance in comparison to other optimizers.

```
optimizer = 'adam'
```

4.2.8 Categorical Cross-entropy

Categorical cross-entropy is a loss function which is generally used for multi classification tasks. For example, this function can be used where an image/data can belong to only one of the several classes present. The model is used to decide that the image/data belongs to which class. This function is designed to assess the difference between two probability distributions. So, for a given image/data, the possibility of it being in a certain class must be 1, and for the rest classes, the probability would be 0. As we sue softmax function and get the probabilities of different classes, we need to then get the amount of loss each class generates. This is used to find out the loss for each class. Model weights during training are adjusted with the help of categorical cross-entropy. A model is most efficient with minimum loss. Thus, the weights are adjusted such that the cross-entropy loss is minimum. A cross-entropy loss of 0 indicates a perfect model.

```
loss = ‘categorical_crossentropy’
```

The above parameters together are used to construct a model that works and predicts based on the dataset it is trained on. A complete code syntax for constructing ResNet50 model is as shown below:

```
base_model = ResNet50(include_top=False, weights = 'imagenet')

x = base_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(1024,activation = 'relu')(x)

predictions = Dense(train_generator.num_classes,
activation='softmax')(x)

model = Model(inputs=base_model.input,outputs=predictions)
```

Chapter 5

Results and Conclusion

5.1 Results

After defining the model, training of the model starts with the Data Generators made earlier. All the three trained models are saved as '.h5' file in the same directory so that these models can be used in future to make predictions based on the training they received. The h5 format is nothing but Hierarchical Data Format version 5 that allows to store complete details about a model including the weights and model configuration in a single file.Once, the training of each of the models is completed, it was found that the accuracy and performance of ResNet50 were the best among the three models compared. Below is the graphical representation of the confusion matrix after training the models.

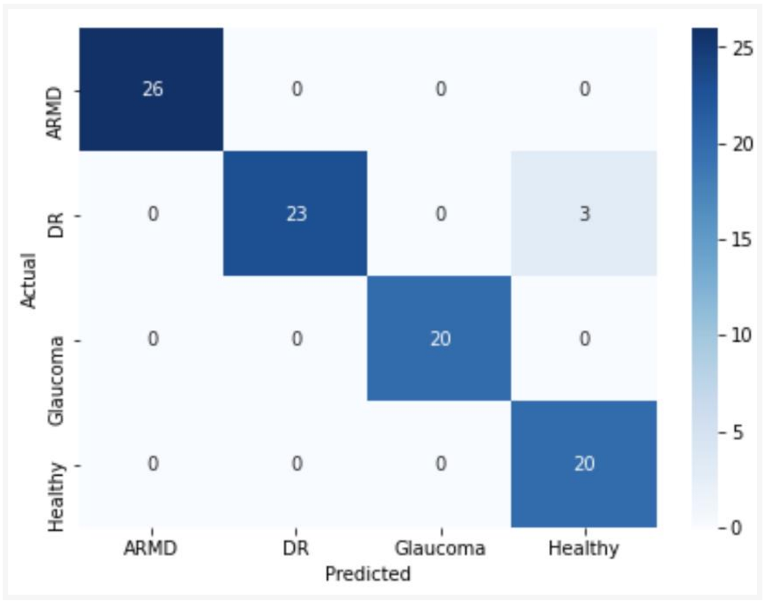


Fig.5.1: Confusion matrix obtained using ResNet50

It can be observed that the accuracy of specific diseases is unity and only for DR it come out to be 88.46% which is best as compared to other models. Hence, this is the finalized model that will be used for final testing that will be done on random fundus images.

The final test is done on random input images that can be healthy or diseased. All the images that need to be tested and prediction is to be made must be stored in a single folder. The path of this folder can be provided in the program for further operations. The trained models which were saved earlier as h5 file are now loaded inside the program.

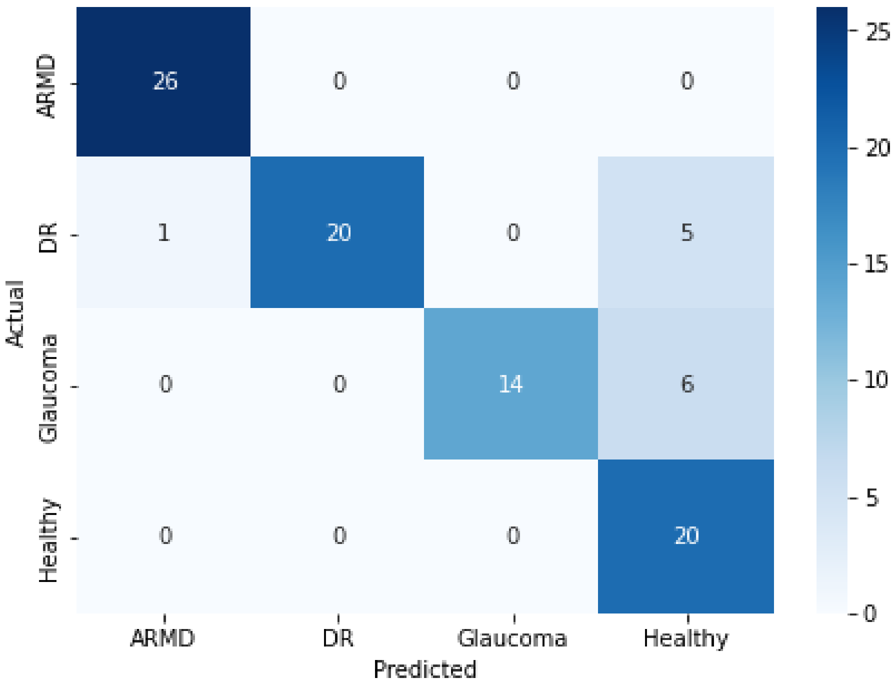


Fig.5.2: Confusion matrix obtained using DenseNet201

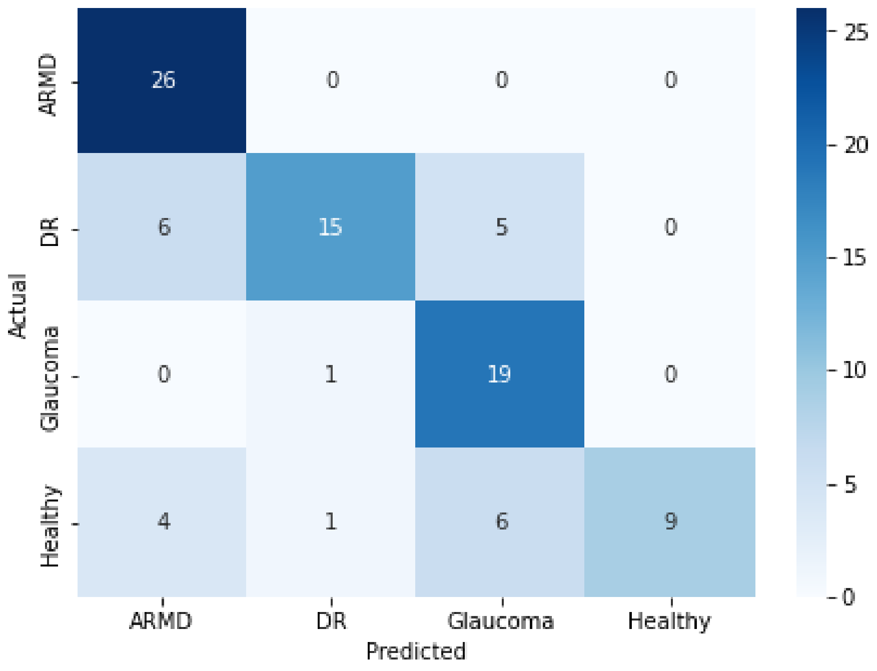


Fig.5.3: Confusion matrix obtained using InceptionV3

5.2 Conclusion

The project is an effort to automate and simplify the process of ophthalmic disease recognition using various sophisticated techniques. This project aims to develop a model which can accurately predict the disease based on the fundus images. As the fundus images may sometimes have the issue of low light or the angle from which the retina is captured can vary, we need to process the data we acquire in order to enhance abnormalities in the image. So we use some processing techniques such as circular crop, gaussian blur, etc. After preprocessing the data, the data is then fed to the models which work on the basis of supervised learning. The model is fed with many images with different labels corresponding to various diseases. The model then finds patterns in them and assigns weights to all the layers based on its learning which is used to predict the output.

In this project, we have compared 3 models namely ResNet50, DenseNet201 and InceptionV3 for their ability to detect various diseases. The diseases include Glaucoma, Diabetic Retinopathy (DR), Age Related Macular Degeneration (ARMD). Each model was first trained for the same set of images. We have provided the same images to each model as we need to compare the models based on their performance. Then based on the results of their accuracies, while every model gave us a good result in terms of accuracy, we can conclude that among the three models, ResNet50 gave better accuracies than the other two followed by DenseNet201 and then InceptionV3. We found out that due to backpropagation, the ResNet model tends to train faster and better in comparison to other models. The purpose of selecting these models for comparison was that the architecture of every model is different and the time required for the model to train on a given set of data is also different.

Though the ResNet50 model works fine with our requirements, for more accuracy and bigger datasets, we can go for higher end models of ResNet such as ResNet101 which consists of 101 layers. Similarly, DenseNet and Inception also have different models which could be used depending on the requirements.

5.3 Scope

This project is only a small part of the tremendous variations which can be found and worked upon when it comes to disease recognition. There are many diseases that can be detected from examining the fundus images alone. Not only are there many diseases, but also their levels of severity. Thus, this multi-classification approach that has been adopted by us can be widened to include more diseases along with their complexity. By giving a general idea about the severity of the disease and the precautions that need to be taken, we can create awareness amongst patients suffering from such diseases so that they can start their treatment before it is too late.

At the same time, with limited access to authentic data and an equally limited computational power, we can show only a glimpse of the entire picture. With more advanced technologies and much more authentic data, softwares which detects such diseases flawlessly can be developed. These softwares can easily detect various different types of diseases and would eventually reduce professional work. These can not only aid professionals make better informed decisions, but also help the common man understand more about their situation without worrying about multiple opinions from professionals.

For immediate ideas, we could develop a web application where the user inputs the retinal image it wants to check. Then the algorithm would predict the disease and provide some information regarding the disease such as the severity of the disease, precautions to be taken, etc. This would be convenient for many people as there is no need to physically go to a doctor and also by the information provided, you can take some precautions early.

References

1. M. H. Sarhan et al., "Machine learning techniques for ophthalmic data processing: A review", IEEE J. Biomed. Health Inform., vol. 24, no. 12, pp. 3338-3350, Dec. 2020.
2. Abràmoff, M.D. & Garvin, Mona & Sonka, Milan. (2010). Retinal Imaging and Image Analysis. Biomedical Engineering, IEEE Reviews in. 3. 169 - 208. 10.1109/RBME.2010.2084567.
3. C. Wang et al.: Pulmonary Image Classification Based on Inception-v3 Transfer Learning Model. IEEE Access, vol. 7, pp. 146533-146541, 2019, doi: 10.1109/ACCESS.2019.2946000.
4. Zhao, Chen & Shuai, Renjun & Ma, Li & Liu, Wenjia & Hu, Die & Wu, Menglin: Dermoscopy Image Classification Based on StyleGAN and DenseNet201. IEEE Access. PP.1-1. 10.1109/ACCESS.2021.3049600
5. https://iq.opengenus.org/resnet50-architecture/
6. https://medium.com/analytics-vidhya/deep-dive-into-resnets-eb4ec48dfcb0
7. https://neoretina.com/blog/diabetic-retinopathy-can-it-be-reversed/
8. https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/
9. https://deepmind.com/blog/article/Using_ai_to_predict_retinal_disease_progression

turnitin™ **Similarity Report ID:** oid:9832:16177581

PAPER NAME

final report.docx

WORD COUNT	CHARACTER COUNT
7953 Words	42948 Characters
PAGE COUNT	FILE SIZE
40 Pages	2.2MB
SUBMISSION DATE	REPORT DATE
Apr 21, 2022 2:05 PM GMT+5:30	Apr 21, 2022 2:06 PM GMT+5:30

● 18% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 13% Internet database
- 7% Publications database
- Crossref database
- Crossref Posted Content database
- 14% Submitted Works database

● Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 30 words)