Prediction of Glaucoma from Fundus images leveraging Transfer Learning in Deep Neural Network

by

Sayem Mohammad Ismail
16321050
Md. Sajjad Hossain
16321062
Irina Sobhan
16121133

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

Department of Electrical and Electronic Engineering
Brac University
June 2020

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:

Sayem Mohammad Ismail
16321050

Md. Sajjad Hossain
16321062

Irina Sobhan
16121133

Approval

The thesis/project titled "Prediction of Glaucoma from Fundus images leveraging Transfer Learning in Deep Neural Network" submitted by

1. Sayem Mohammad Ismail (16321050)
2. Md. Sajjad Hossain (16321062)
3. Irina Sobhan (16121133)

Of Spring, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Electrical and Electronic Engineering on July 13, 2020.

Examining Committee:

Supervisor:
(Member)

Dr. Mohammed Belal Hossain Bhuian
Associate Professor
Department of Electrical and Electronic Engineering
Brac University

Co-supervisor:
(Member)

Dr. Md. Ashraful Alam
Assistant Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Abu S.M. Mohsin
Assistant Professor
Department of Electrical and Electronic Engineering
Brac University

Head of Department:
(Chair)

Dr. Shahidul Islam Khan
Professor
Department of Electrical and Electronic Engineering
Brac University

Acknowledgement

All praise to the Great Allah for whom our thesis have been completed without any major interruption. We are extremely grateful to our Supervisor Associate Professor Dr. Mohammed Belal Hossain Bhuian and Co-supervisor Assistant Professor Dr. Md. Ashraful Alam for their kind support and advice in our work. Without their continuous support this thesis would not have been possible. We are grateful to Mr. Md Tanzim Reza for his love and support for us in learning Convolutional Neural Networks, Deep Learning and Transfer Learning. He helped us whenever we needed help. We pay our gratitude for him. And finally we pay our love, respect to our parents. With their kindness, support and prayer we are now on the verge of our graduation.

Abstract

Transfer learning techniques in deep learning is nowadays a raising and promising field of research and a tool for Artificial Intelligence with a lot of prospects. Our goal is to predict Glaucoma from fundus images to help the diagnosis procedure of Glaucoma, a public health hazard, at an early stage. In this research, we propose a transfer leaning methodology, creating four models with four pre-trained CNNs implemented separately in each of the models, trained and tested for detecting Glaucoma from fundus images. We have used VGG19, ResNet50, DenseNet121 and InceptionV3 for our transfer learning models, with the fine-tuning approach to ensure better learning performance on our dataset of labelled fundus images. Fine-tuning is done keeping all the layers of pre-trained CNN trainable on the fundus image dataset, and applying the classic method of adding a customized classifier. All the four transfer leaning models are Deep Neural Networks carrying deep hidden layers as the pre-trained CNN implemented. Deep learning application on Biomedical field is itself a challenge to work with due to shortage of labeled data. Thus transfer learning is found very effective in working with a small image data set to predict Glaucoma. Our proposed models built with VGG19, ResNet50, DenseNet121 and InceptionV3 deliver test accuracy of 94.75%, 96.5%, 92.5%, 91.75%. In order to achieve such accuracy in biomedical application, transfer of knowledge of features learned of pre-trained CNNs gave a competitive edge on initialization of parameters. We present comparison amongst the models proposed and the ResNet50 built model gives the best performance.

Keywords: Glaucoma; Artificial Intelligence ; Fundus images; CNN; Transfer leaning; Prediction; Deep learning; VGG19; ResNet50; DenseNet121; InceptionV3; Fine-tuning

Table of Contents

List of Figures

List of Tables

Nomenclature

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

$(C/D)ratio$ Cup to Disk ratio: ratio of the distance between optic disc center and optic nerve head to diameter of the optic disk

ANN Artificial Neural Network

CNN Convolutional Neural Network

DNN Deep Neural Network

$ILSVRC$ ImageNet Large Scale Visual Recognition Challenge

IOP Intraocular pressure

NN Neural Network

$ReLU$ Rectified Linear Units

Chapter 1

Introduction

1.1 Motivation

Among all the eye diseases glaucoma is a leading cause of blindness due to damage of the optic nerve at the back of the eyeball. If Glaucoma is detected at a primary stage, then it might be prevented from causing severe damage to the eyes and also the loss of vision can be slowed down or prevented. Our eyesight is one of the most important senses that help us perceive 80% of the things around us. Glaucoma is a disease, that can occur at any age but most commonly it is found among the elderly people. Moreover, glaucoma is a chronic disease that develops slowly which might not come to our notice and is usually identified when the condition gets severe. Therefore keeping in mind, the importance of eye health, we have come up with the idea of eye diagnosis that would help the medical practitioners to detect glaucoma at a primary stage. After a lot of research we found out that a few remarkable research works have been done based on this glaucoma detection. Our main aim was to try some different methods with effective results that would prove to be useful for the doctors. Hence, with the promising method of transfer learning in deep learning, we have developed deep neural network models that will detect Glaucoma and tried comparing all the results in order to find out which model gives the best result.

1.2 Literature review

L. Li *etal.*(2019) worked on Attention Based Glaucoma Detection by using a CNN model. They have proposed AG-CNN, a new deep learning method, for automatic glaucoma detection and pathological area localization upon fundus images and attention maps. Furthusmore, they Large-scale attention based glaucoma (LAG) database, with sizeable number of fundus images and corresponding labes attentiona maps. Each of the fundus image was diagnosed by qualified glaucoma specialists. They then designed a new AG-CNN structure which included an attention-prediction, a pathological area localisation and glaucoma classification subnets. They further conducted another experiment to capture the attention regions of the ophthalmologists in glaucoma diagnosis. After conducting all the experiments they got result of accuracy around 95.3% with their AG-CNN approach [33].

Similarly, Guangzhou *etal.* (2019) also worked on glaucoma detection with Machine learning based on three-dimensional optical coherence tomography (OCT) data and fundus images. For their study, they collected their data from 208 glaucomatous and 149 healthy eyes enrolled. However, they used the fundus images, acquired as colored, in grayscale format. On top of that, the number of sample they were using is significantly small. In their research, they adopted the CNN architecture VGG19. They did transfer learning of CNN with fine tuning using input images as described before. Performing data augmentation and dropout they built 5 classification models with VGG19 for each kind of images, one for the fundus images input and rest four of them developed with four different type of OCT data. They found 0.940 or 94.0% accuracy for the CNN taking fundus images input, and similar for the rest of the OCT input models. [26].

Perdomo *etal.*(2018) proposed a multi-stage deep learning model for glaucoma diagnosis with strategy of curriculum learning, where model is sequentially trained in solving tasks off gaining complexity. Such technique helps out when the data is small in number, which is almost always the case in biomedical research. They performed stages of optic disc segmentation, morphometric features prediction from segmentations, and disease level prediction. They acheived an accuracy of 89.4% [21].

1.3 Overview

This paper consists of in total 7 chapters where chapter 2 discusses the Glaucoma and the traditional procedures of diagnosis of Glaucoma. Chapter 3 contains the acquisition of dataset. Then, chapter 4 discusses the fundamentals of deep learning and transfer learning and their procedure. After that chapter 5 contains the proposed methodology of our research. Then chapter 6 discusses the results found out and analysis of our result and proposed models. Furthermore, chapter 7 contains the scope for future works and concluding remarks.

Chapter 2

Glaucoma and its diagnosis

2.1 Glaucoma

Glaucoma is a chronic eye disease which may lead to permenant loss of vision. It is believed that about 4.5 million individuals worldwide have lost their vision being affected by glaucoma [38]. Glaucoma damages optic nerve inside the eyes. For good vision it is important to have healthy optic nerve. An abnormally high pressure on eyes known as intraocular pressure(pressure of fluid within the eyeball) may cause optic nerve to be damaged [39]. When the damage worsens, people may suffer loss of vision. This whole scenario is known as glaucoma.

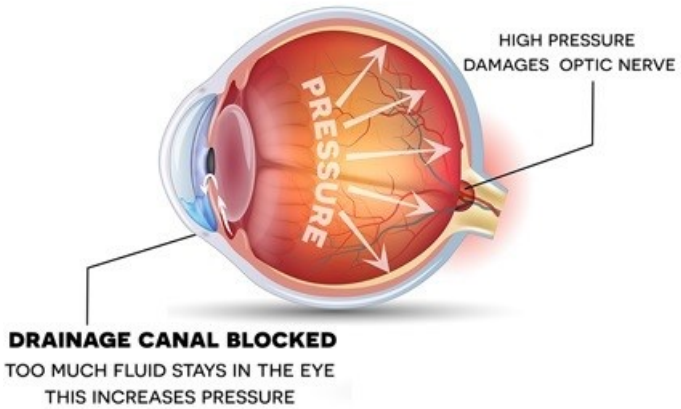


Figure 2.1: Glaucoma [38]

Glaucoma counts as one of the major causes of vision loss for people relatively older, however, it may affect young individuals as well [19]. Usual forms of glaucoma generally do not display signs at early stages, and it does not cause any pain either. The condition inside of an eye gets worsen gradually and the optic nerve gradually damages. Hence, regular eye diagnosis is needed so that the changes of an eye can be noticed at an early stage. There are usually some changes that are noticeable between a healthy eye and an eye affected by glaucoma. In normal eye the optic nerve is seen to be thin and straight but inside of a glaucoma affected eye, some differences are noticed. The optic nerve gets swollen and it displaces from its initial

position and it does not remain as straight as before.

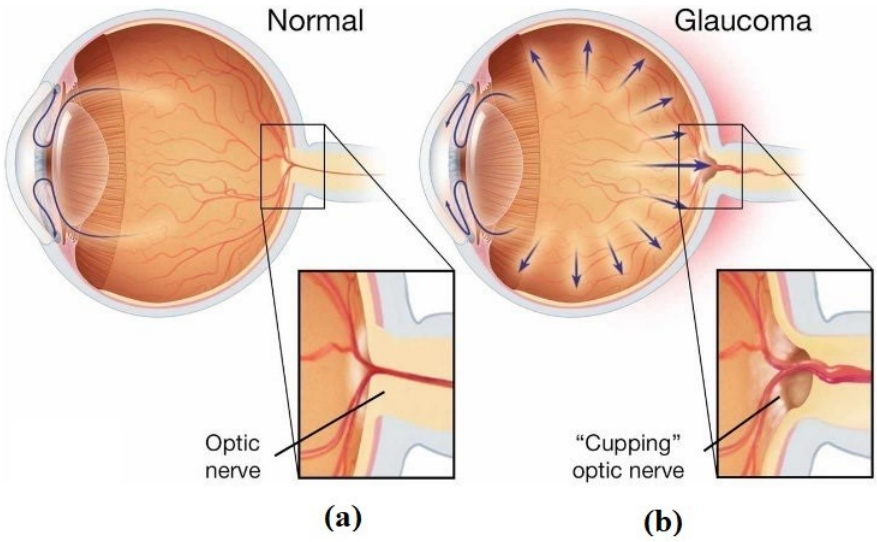


Figure 2.2: Normal and glaucoma affected eyes. [43]

2.1.1 Glaucoma categories

Among many, there are mainly two categories of glaucoma[34]:
Open-angle glaucoma:
This category of glaucoma is the most usual one. In this type, the fluid inside our eyes flows abnormally.

Angle-closure glaucoma:
Such type of glaucoma is usually found in Asia region. Here, the drain space in between the iris of the eye and the cornea of the eye gets narrower. This type of glaucoma causes sudden buildup of pressure in our eyes.

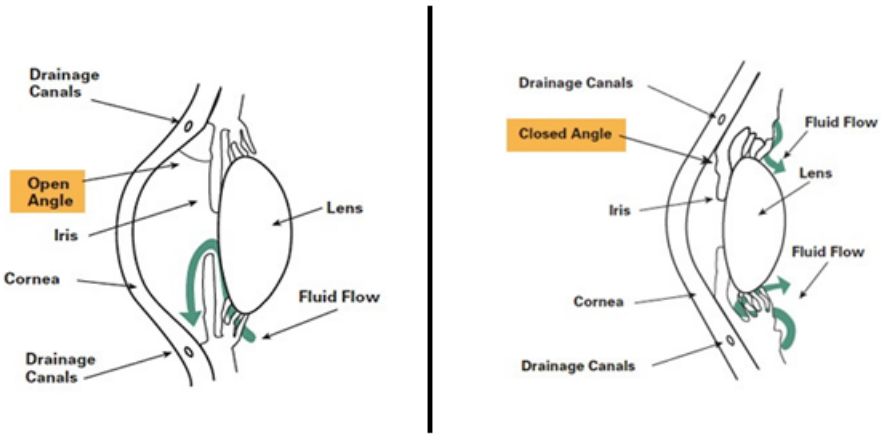


Figure 2.3: open-angle glaucoma and angle-closure glaucoma [18]

There are some other categories of glaucoma as well. A regular diagnosis is important so that early detection of glaucoma is possible and Loss in vision can be slowed

down, or stopped.

2.1.2 Risk Factors

Glaucoma is such a disease which does not show some major symptoms in its earlier stage. Later on when the condition gets worse than before then it shows some symptoms. There are some characteristics for any human which can increase the chance of being affected by glaucoma. These are the risks factors which are mostly responsible for the development of glaucoma [40]:

- Age: People who are over 60 years old are at high risk for getting affected by glaucoma. On the other hand, African Americans are at high risk when they are over 40 years old. Actually, when any person gets old the risk also increases with the increase of his age.

- Family history of glaucoma: A man will have a high risk of getting affected by glaucoma if any of his family members were affected by it.

- Other diseases: Medical studies has shown that the risk of being affected by glaucoma is higher for the people having diabetes or high blood pressure or heart conditions.

- Physical injuries of eyes: If someone gets hit in the eye then that may cause severe pressure inside the eye which may lead to the development of glaucoma. Internal damage inside the eye can also increase such pressure and can cause glaucoma development.

- Corticosteroid: Using corticosteroids (a class of drug that lowers inflammation in the body) for a long period of time may create risk of being glaucoma affected which may turn into serious condition later on.

- Other eye related risk factors: There may have some particular eye features like a thinner cornea or sensitivity in optic nerve can also lead to glaucoma development inside one's eye. Some others conditions like retinal detachment, eye tumors and eye inflammations can also increase the chance of glaucoma development and these can trigger it very fast.

2.2 Diagnosis of Glaucoma

A regular diagnosis is needed to know about the affection of glaucoma because when an eye gets affected by glaucoma, changes occurs internally inside an eye. For detecting glaucoma different approaches are used.

One approach is the measurement of intraocular pressure (IOP) [27]. This test is known as tonometry. During this test sometimes eye drops are used to make the cornea less sensitive. IOP readings that is normal is typically less than 21 mmHg (millimeters mercury). This is measured of force exerted on a defined area. When the IOP readings gets higher, the risk of getting affected by glaucoma increases.

There is another approach that include the use of advanced imaging to establish reference images and measurement of the optic nerves of the eyes and other orders inside the eyes. The images are taken repeatedly and the measurement of the optic nerve are observed. If changes are observed then there is a possibility of glaucoma affection.
Nowadays, all the above mentioned approaches are used for the detection of glaucoma.

2.3 Detection of Glaucoma by Fundus Image

The most recent approach for glaucoma detection is done by using fundus images. In this section, we will discuss about what fundus images are and how it is used to figure out glaucoma with all the features that we get from fundus images.

2.3.1 What is Fundus Image?

Fundus images are photographs taken by color fundus camera. In order to track the existence of conditions and control their improvement over time, fundus camera takes color photographs of the state of the eye's interior surface. Fundus camera is eqquiped with sophisticated microscope where the power is low, and it can take images of retina documenting several features, for example diabetic retinopathy, macular degeneration[37] etc.

2.3.2 How Fundus images are used to detect Glaucoma

When glaucoma starts to grow inside an eye, the changes are so slow that it does not show any major changes suddenly. Moreover, the internal structure of an eye is complicated. So, the changes are too acute to observe manually. There develops some changes in a glaucoma affected eye and it creates some differences from a normal eye.

Fundus images show the internal condition of an eye acutely. It is easy to notice the changes from the fundus images of a glaucoma affected eye. When the recent fundus images of an affected eye are compared with the fundus images of that eye which were captured at normal condition, the changes are observed depending on some features. Some of these features are like (C/D) ratio[4] etc. Such features in a glaucoma affected eye and in a normal eye vary from each other.

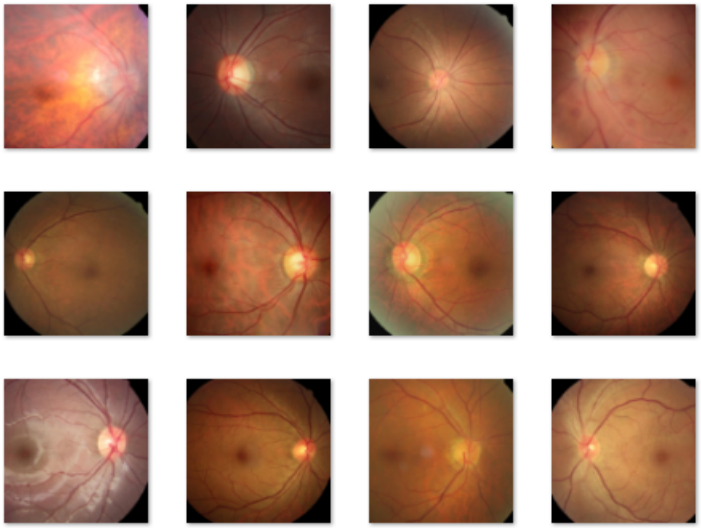


Figure 2.4: Fundus image. Source: dataset

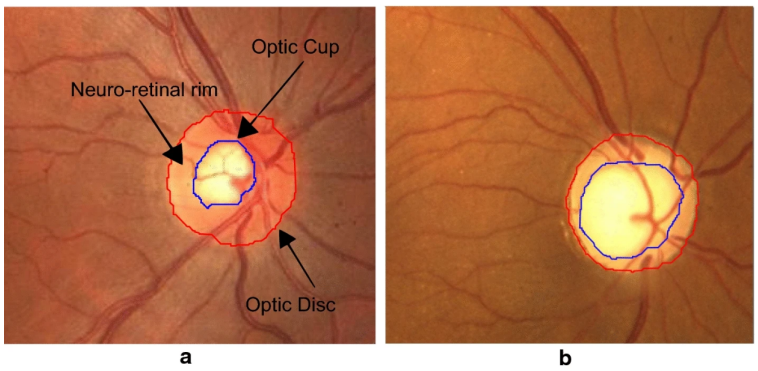


Figure 2.5: Changes inside an eye from normal condition to glaucoma affected condition. [29]

Some glaucoma affected fundus images with their corresponding normal condition are shown below. These fundus images are captured from very close to one's eye. From these images, it is seen clearly that the changes are explicit.

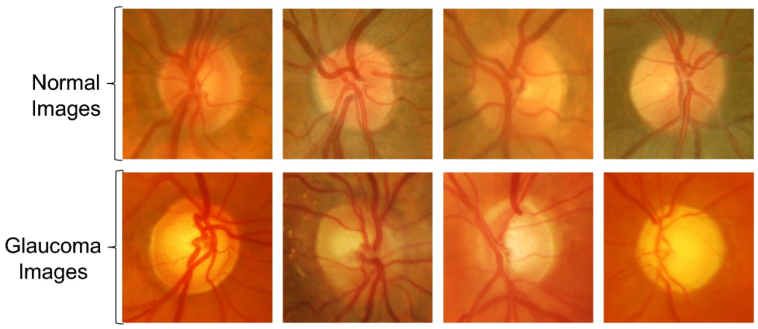


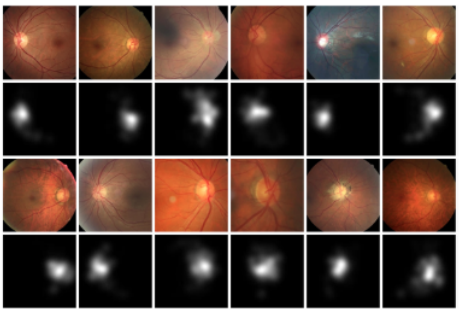
Figure 2.6: Changes inside an eye from normal condition to glaucoma affected condition. [29]

When glaucoma starts to develop inside the eye then these ratios changes from the previous condition. Hence, these changes of features are considered to find out glaucoma by using various machine learning models. Nowadays, image processing is widely used in different arenas of our life. In deep learning, these features are extracted automatically.

Chapter 3

Acquisition of dataset

Data is the first and foremost ingredient for any deep learning application. The problem comes when the data is real time. Creating and labeled data in real life is enormously time consuming, effort demanding and expensive as well. Especially, in biomedical applications getting labeled data is even more difficult, thus resulting in datasets of relatively short amount of data to work with, if compared to other fields of data. Which indeed justifies the necessity of transfer learning.
In this experiment LAG-database [33] has been used for detecting Glaucoma using transfer learning method of deep learning.



(b)

Figure 3.1: sample data from LAG-Database [33]

The acquisition of LAG-Database [33] is done on academic ground, that contain 4854 fundus images along with corresponding labels as well as attention maps belonging to two classes, where one of them named as "Suspicious Glaucoma" and the other one as "Non Glaucoma". The fundus image data from the class "Suspicious Glaucoma" carry labels 1 and the class "Non Glaucoma" carry labels 0. Demonstrated in table 3.1 below:

The attention maps provided in the dataset, however, haven't been used in our work, rather we used the labeled fundus images only.

Glaucoma and non-glaucoma fundus Images

Class	Label	Fundus images	Attention maps
Suspicious Glaucoma	1	1711	1711
Non Glaucoma	0	3143	3143

Table 3.1: Distribution of image data with labels in LAG-database[33]

Figure 3.2: Non Glaucoma

Figure 3.2 represents fundus images from the "Non Glaucoma" class. They are captured from healthy eyes. Whereas, the figure 3.3 represents fundus images from class "Suspicious Glaucoma" captured from Glaucoma affected eyes.

Figure 3.3: Glaucoma affected

Chapter 4

Fundamentals of Deep Learning and Transfer Learning

Deep learning is generally referred as a section of machine learning, that deals with algorithms, which was developed in inspiration from functionality and layout of human brain, known as artificial neural networks [28]. For deep learning, the neural network is called as deep neural network.
Learning theory for deep learning are generally recognized as two categories and they are -

- Supervised: Where the data contain its corresponding label.
- Unsupervised: Where the data doesn't contain its corresponding label.

Supervised learning are such kind of learning where the all the data fed into the algorithm is labeled, whereas the unsupervised learning does not require a labeled dataset.

4.1 Deep Neural Networks or DNN

An usual neural network is made up with several basic units, which are also known as neurons, connected to each other in a particular design and able generate and hold sequential activation that are real-valued [12]. The neurons in such artificial neural network are also called as units, shown in figure 4.1 are organized in layers.
The input layer is the first layer that takes input from the data given into the neural network. Similarly the layer provides output is known as the output layer. Both the input and output layers may contain number of units, however, the output layer generally gets as many units as there are output classes. Deep Neural Network has got input layer, output layer and number of hidden layers confined in between them, demonstrated in figure 4.2.

Constructing a neuron or unit in deep neural network is done with some parameters known as weights and biases, which indeed are the basic working principle for a deep neural network.The assigned weights of the connection is multiplied to its corresponding input and then added up resulting into a weighted sum, on which the activation function implies and produces the activation for that neuron, which

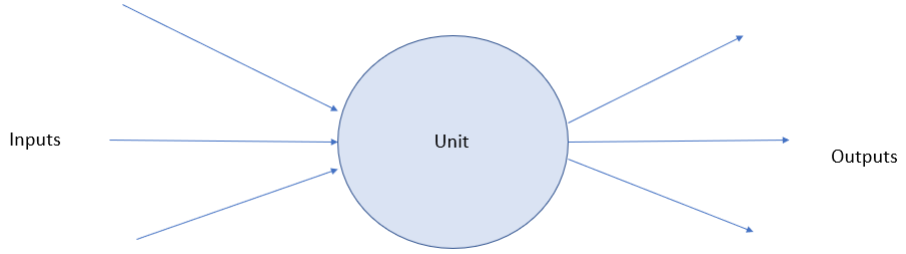


Figure 4.1: A unit or neuron in an artificial neural network(ANN)

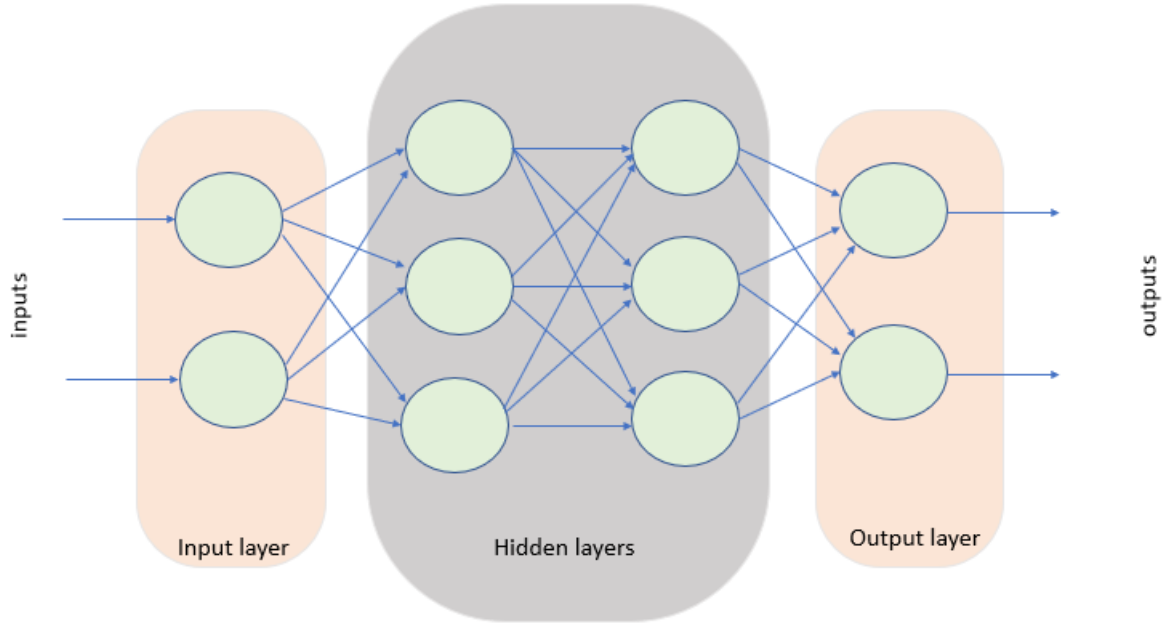


Figure 4.2: Two hidden layers confined in between input and output layers

is indeed the output of the neuron denoted by y. A demonstration is given in the figure 4.3. The activation of this neuron is,

$$a = f\left(\sum_{i=0}^{2} x_i w_i + b\right) \tag{4.1}$$

This activation a is the output y of this neuron. For a neural network as in figure 4.4 built up with units like figure 4.3, the equation 4.1 converts into matrix domain.

$$\begin{pmatrix} a1 \\ a2 \\ a3 \end{pmatrix} = f\left(\begin{pmatrix} w11 & w12 & w13 \\ w21 & w22 & w23 \\ w31 & w32 & w33 \end{pmatrix}\begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix} + \begin{pmatrix} b1 \\ b2 \\ b3 \end{pmatrix}\right) \tag{4.2}$$

Taking the wight matrix as W, the input vector as X, the bias vector as B, and the activation vector as A,the equation 4.2 is as follows [35] ,

$$A = f\left(WX + B\right) \tag{4.3}$$

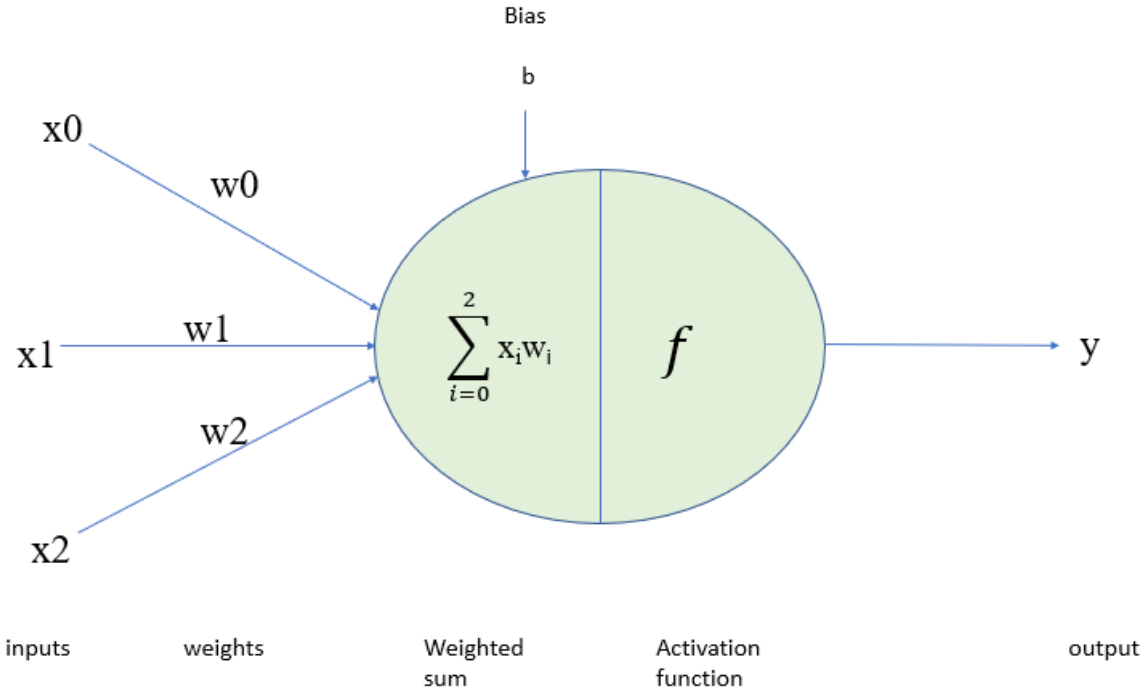


Figure 4.3: A unit or neuron in neural network

All other layers of units in neural networks are activated in such manner, where the activation function f is generally Sigmoid.

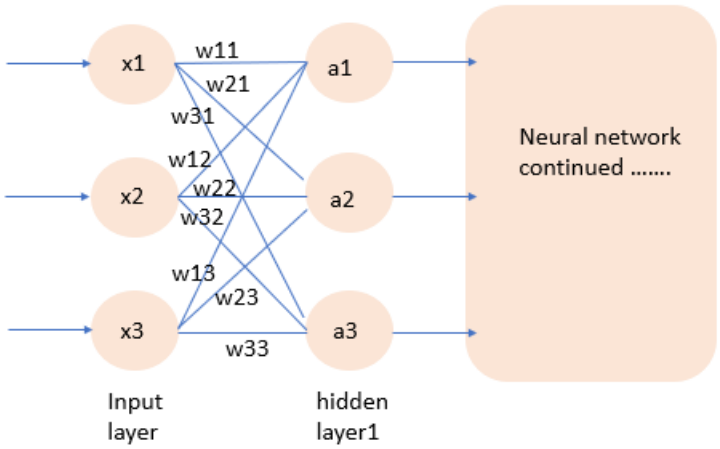


Figure 4.4: Neural network with parameters shown

These weights are randomly initialized at the first place and then through a number of iterations in the training process the weights are gradually updated and learned, so that it can predict new data at a satisfactory level, which it had never seen before. The neural network is evaluated through a cost function. This function tells us if the weights and biases are well chosen, or if they must change. The target

is finding the global minimum for the cost function. To minimize cost, a method called backpropagation is used combined with optimization algorithm. As the name of the method suggests, the error is propagated backwards in the network, and it simultaneously updates the biases and weights [42]. The figure 4.5 is showing the

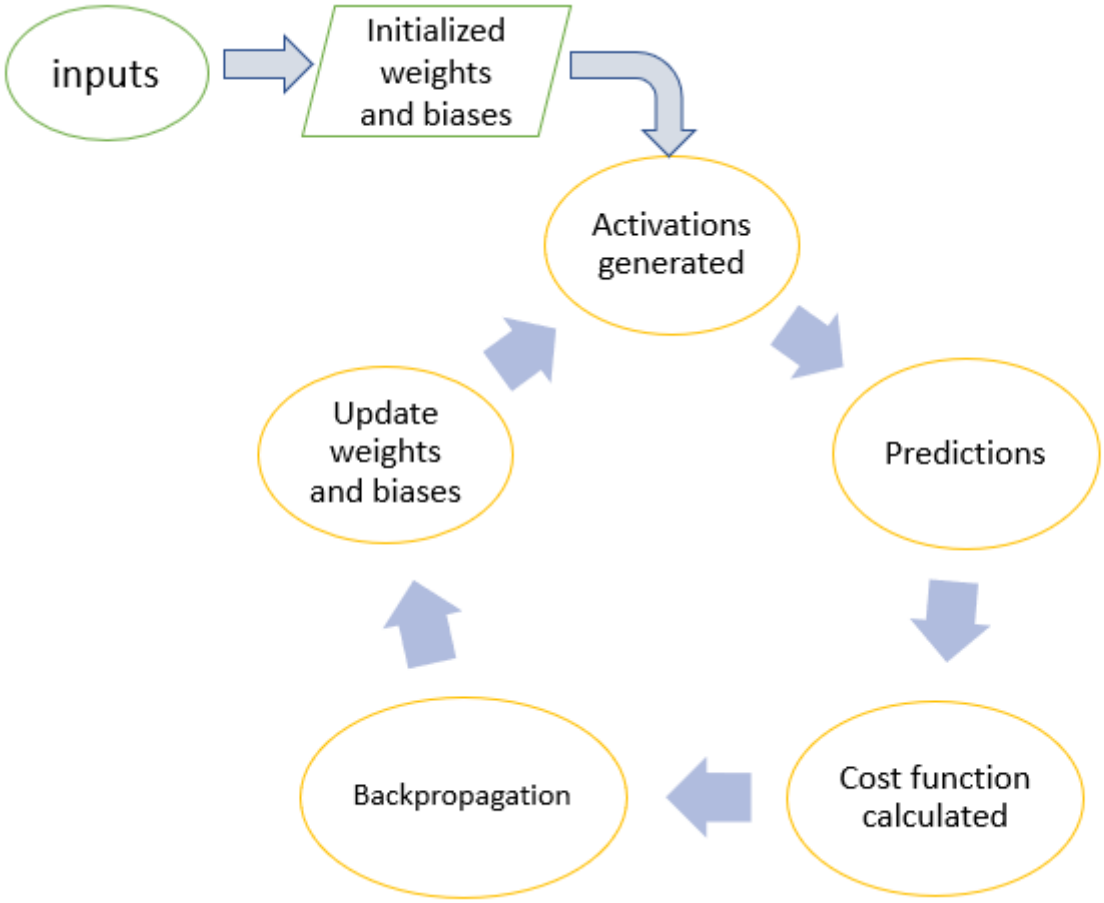


Figure 4.5: The flow of action in a neural network throughout the learning process

flowchart for the mechanism for a deep neural network to learn to classify data of its target domain.

4.2 Convolutional Neural Networks or CNNs

In convolutional neural networks the convolution operation is performed by the layers hidden, on ground of which features are learnt by the neural network and the necessity of manual extraction of features is relieved. CNN performs well for image data. [16].

An image is a two dimensional matrix of its pixels, as in figure 4.6, and that is what CNN takes care of as an input to process in image classifications.

CNN thus take in an image data as an array of numbers and performs number of convolution operation using tools like filters that are relevant and continuously

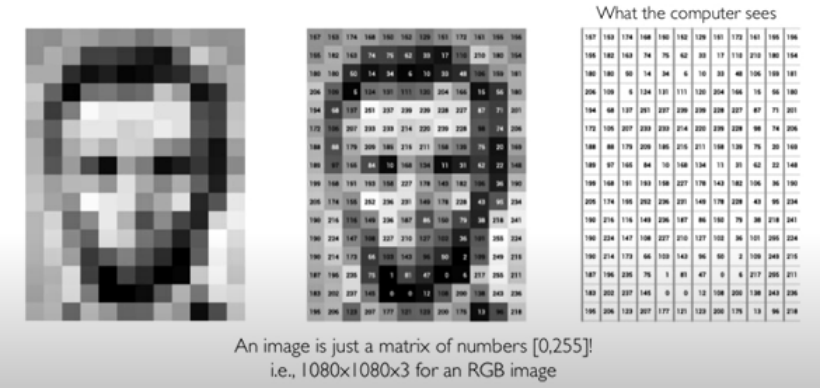


Figure 4.6: Images are numbers [36]

learnt with sophistication through the learning process [22].

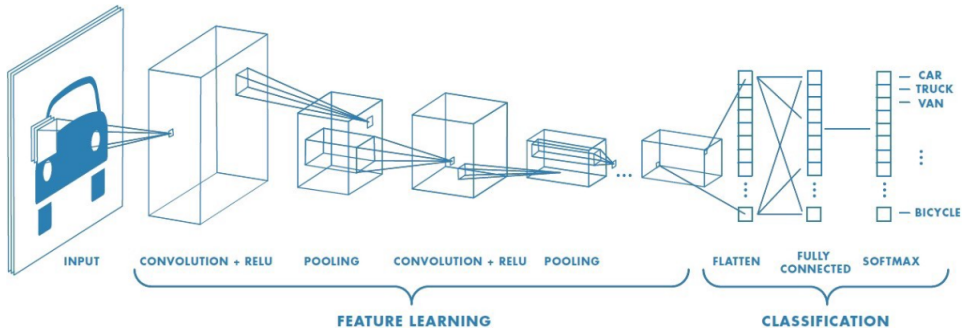


Figure 4.7: A CNN architechture [22]

4.2.1 Segments of CNN architecture

The general architecture of a CNN can be assumed in two segments of action [24], shown in figure 4.7, they are -

- Feature extraction or Feature learning
- Classification or Classifier

Feature extraction or Feature learning
This larger segment of CNN, known as feature extraction or feature learning, is that does the work of learning features present in the images of inputs. To demonstrate, if the image in figure 4.8 represents a house then the doors, windows, steps will be identifying features or if the image is a car then the wheels, the number plate will be identifying features, on the basis of which the image can generally be classified into a group of objects.
Through the training process it picks out hierarchy of features to accurately map the input images in order to learn their features, as in figure 4.9, which eventually leads up to the classifier to use those features maps so that it can predict the class

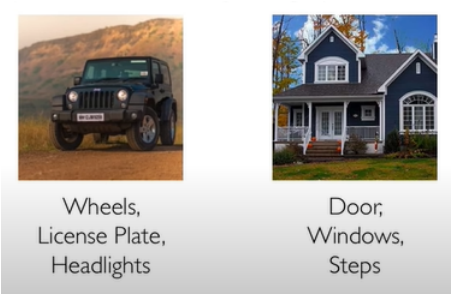


Figure 4.8: Features of from an object in an image [36]

where the data of input belong to.

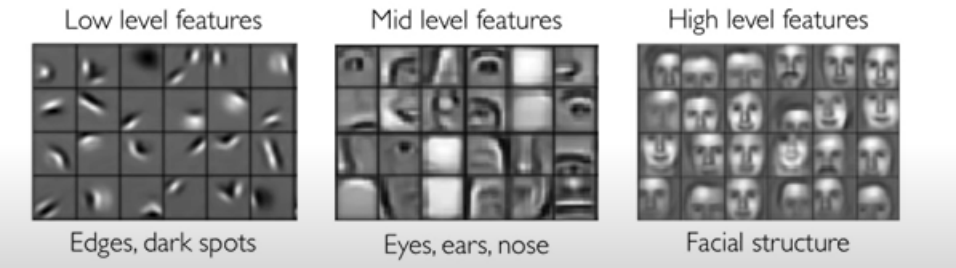


Figure 4.9: Hierarchy of features are demonstrated for an image [36]

This complex work of mapping the features of the input images is done through number of sets of convolutional layers and pooling layers stacked repeatedly [24].

Classification or Classifier

The classification or classifier segment does the actual job of classifying each image or input data into its class of belonging. For example, in context of the figure 4.8, the classifier segment will place the image into the class of either a car or a house by generating a probability distribution of being that element of the corresponding class. In order to generate this classification of data, the classifier segment takes the feature maps, generated by the feature learning segment, as an input. The mechanism shown in figure 4.10

The classifier is built up with dense layers or in other words fully connected layers [24].

4.2.2 Layers and Operations of CNN

The CNN architecture is consisted of three basic type of layers[16] :

- Convolutional
- Pooling
- Fully connected

All these layers of CNN does the essential operations as follows-

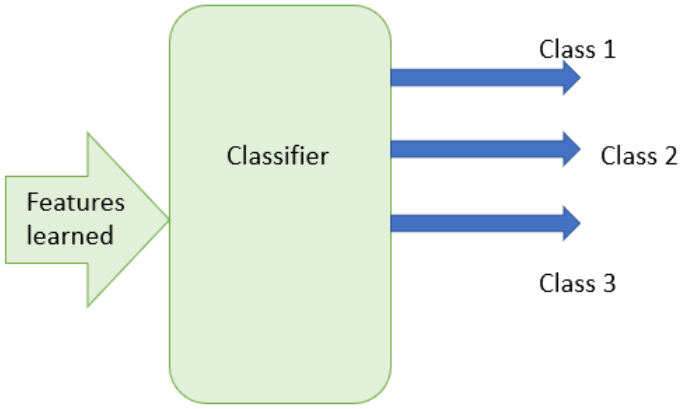


Figure 4.10: Classifier demonstrated

- Convolution operation
- Pooling operation
- Flattening
- Non linear activation function imply
- Optimization operation

Convolutional layer: Convolution operation

The fundamental building block of CNN, the convolutionl layers are responsible for convolution operations of CNN and produce feature maps, that learn the features of the image taken as input, through convolving appropriately learnt filters or kernels with the input arry or tensor, as shown in figure 4.11. A number of feature maps are there in a convolution layer, as shown in figure 4.12, that capture new features and respond to feature hierarchy throughout the neural network from the initial layers to the far edging layers [16], [22], [24].

Pooling layer: Pooling operation

Pooling operation is performed on the feature maps to better extract features and reduce its dimension [24]. Generally the pooling operation is performed as such:

- Max pooling: As output, it provides the maximum value of a patch of numbers from the feature map taken as input [24], figure 4.13.
- Average pooling: As output, it provides the average value of a patch of numbers from the feature map taken as input [22].

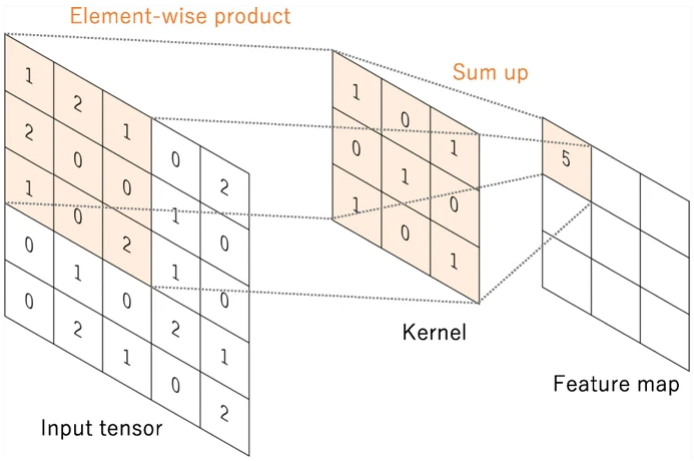


Figure 4.11: A kernel is applied on input tensor, generating a feature map [24]

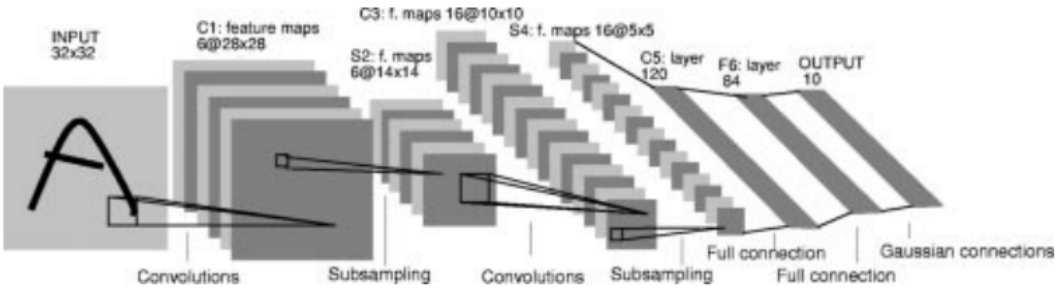


Figure 4.12: Each plane shown is a feature map [1]

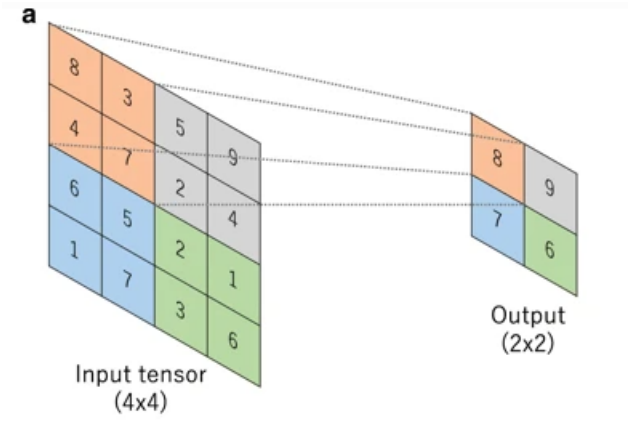


Figure 4.13: Max pooling with a filter of size 22 with a stride of 2

Max pooling and Avarage pooling is compared in 4.14

Flattening layer

Apart from these three basic layers, there is this Flattening layer in CNN that flattens the feature map for the fully connected layers to take as inputs. The flattening operation performed is reshaping the feature maps into vector forms from their original matrix formation, shown in figure 4.15. This operation is very essential because it enables the fully connected layer ahead to work with the newly reshaped feature

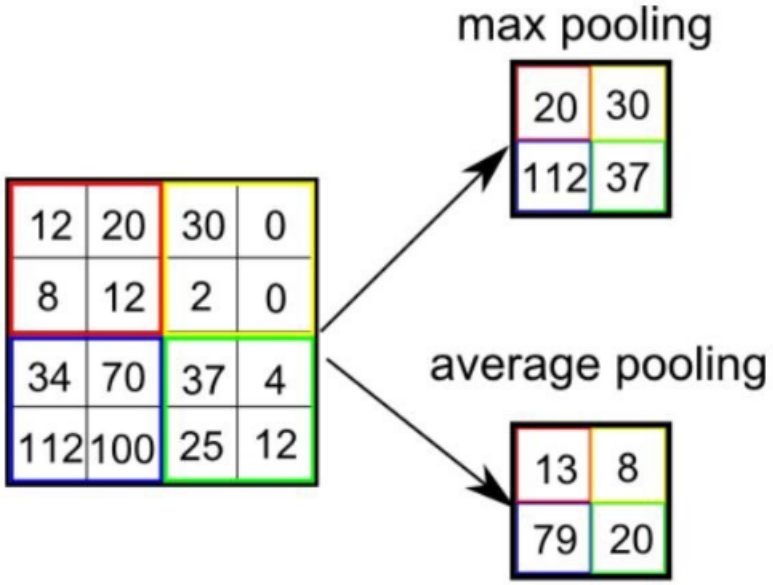


Figure 4.14: Types of pooling [22]

vector [32].

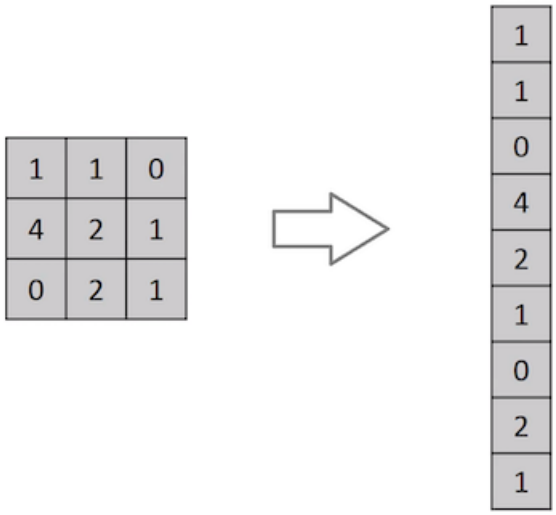


Figure 4.15: matrix to vector flattening [22]

Fully connected layer
In a kind of layers where each units or neurons are connected to all other units or neurons of the subsequent layers are called fully connected or dense layers. Such kind of layers are found useful for all different types of neural networks including CNNs, and constitutes the classifier segment in a CNN and generally becomes the output layer as well [41], [16], as shown in figure 4.16.

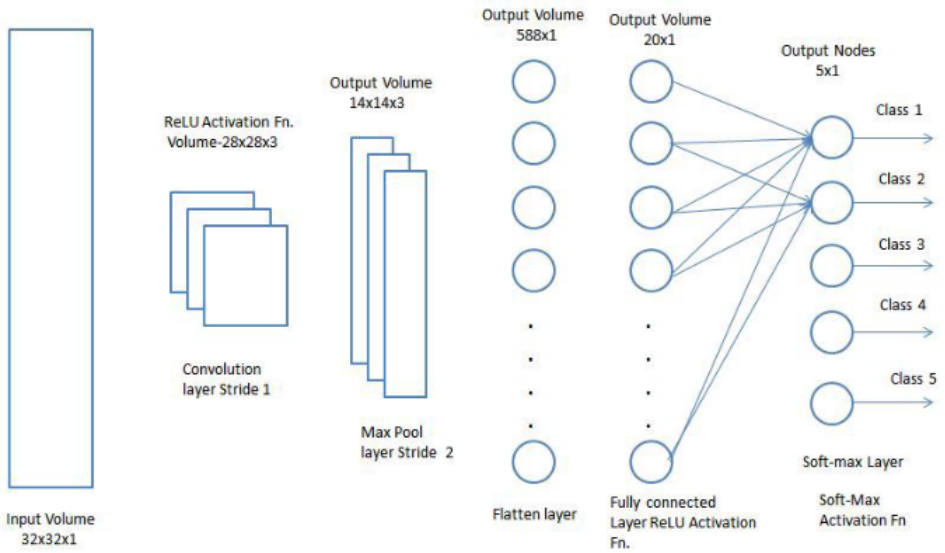


Figure 4.16: Fully connected layers of a CNN leads to classify between classes [22]

Fully connected layers or in other words dense layers take nonlinear activation functions, however, the final output layer of the fully connected layers take Softmax activation.

Activation functions
Activation functions help activate neurons in a layer, by passing the output of convolution operation through non linearity.
Mathematically this is called curve fitting. Three common activation functions are -

- Sigmoid : It is like a step function but it is continuous and differential, which makes it very interesting and important. It's mathematical expression is [35] -

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{4.4}$$

- ReLU: Rectified Linear Units or ReLU are linear for inputs greater than zero [5].

  Definition of ReLU is -

$$f(x) = max(0, x) \tag{4.5}$$

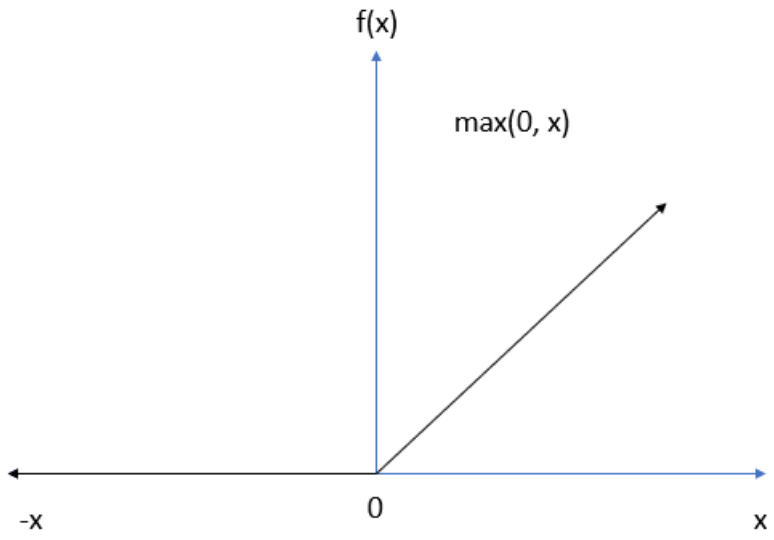


Figure 4.17: Graphical depiction of ReLU

- Softmax: Softmax picks up the maximum value amongst the probabilities and thus is generally used for final classification purposes.

Cost function and Optimization operation

Cost or loss function evaluates the error of the prediction made by the neural network with respect to the real value or label given for that data, so that the error can be reduced gradually throughout the learning process [24].
This cost function defined error between the prediction of the CNN model and the real label of the data, needs to be optimized to a acceptable point where the CNN can produce classification of the data with utmost accuracy it may possibly achieve. Thus, an optimization function optimizes the cost. Adam [7] optimizer is widely used because of its cutting edge efficiency.

Figure 4.18, denotes that, the cost function is minimum for [7] with Adam optimizer comparing other optimizers.

4.3 Transfer Learning

Artificial neural network is inspired from human brain. It is then essential to be motivated from the human learning process and capabilities. Human brain can learn a task in an area and then apply the knowledge into an another task of another area to learn and do that task. For example, if someone learn to ride a bicycle then he or she can easily ride a motorcycle only he or she understands the control mechanism. Here, the knowledge of riding a bicycle is applied or we might say transferred to

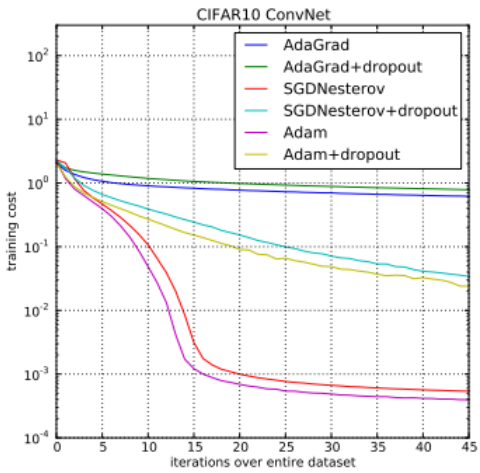


Figure 4.18: Adam is compared to other optimizers [7]

ride the motorcycle, like keeping balance on two wheeler, moving and maintaining directions etc.

Usually, deep learning models are created to learn and perform a specific task. They learn from a dataset of target and performs that task of target. The concept of transfer learning comes into being to transfer the knowledge of such a model into another model building and learning process on another data and task of target,, so that the new model doesn't require the necessity to develop and learn from scratch [23].

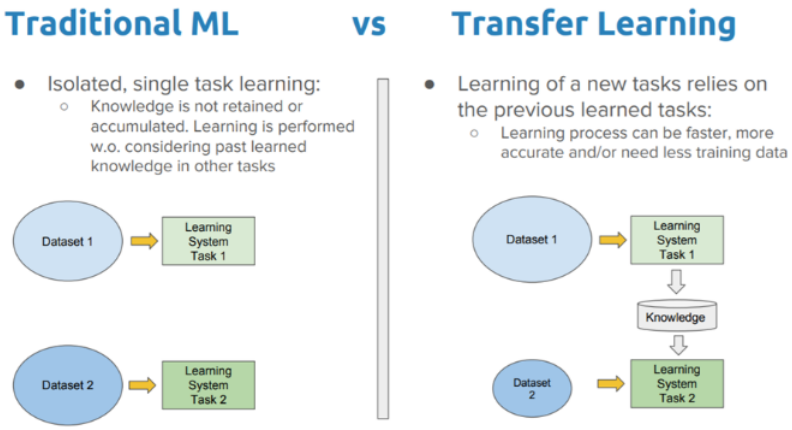


Figure 4.19: Traditional machine learning vs Transfer learning [23]

In figure 4.19, the concept of transfer learning is demonstrated, where knowledge is generally referred to the feature maps, weights and biases etc or in other words all the parameters that has been learned through training process of that model previously trained or usually called a pre-trained model.

Such learnt learnt parameters from a pre-trained model provides a favourable initialization for the new model to learn the data of the target and do the target tasks. This process helps substantially to save time and valuable resources, especially if

there are a relatively small dataset available for the target task to learn on. Pragmatically, most of the sophisticated real world problems do not provide opportunity for a very large amount of labeled data to build and train a complex model. On top of that, creating labelled data is expensive in the real world problems. On the other hand, the task dataset where the model is transferring the knowledge from, may contain a lot of labelled data, for example to say, millions of labelled data. Therefore, leveraging the knowledge from source to kickstart the model for target task dataset is the key [20], [30].

Definition of Transfer Learning Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T , transfer learning aims to help improve the learning of the target predictive function $f_T(.)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$ [6].

4.3.1 Transfer learning techniques

Two popular techniques of transfer learning with CNNs are [46] -

- Feature extraction
- Fine tuning

Feature Extraction:

In this technique, the feature extraction segment of the pre-trained model are not trained on the target data but made to freeze. All the parameters it contain, that was trained on the source dataset, are hence not trainable for the target dataset but only extract features from the data of target task utilizing their knowledge previously learnt and forward them to the trainable classifier that is newly added in most cases to classify the target data for target task [46].

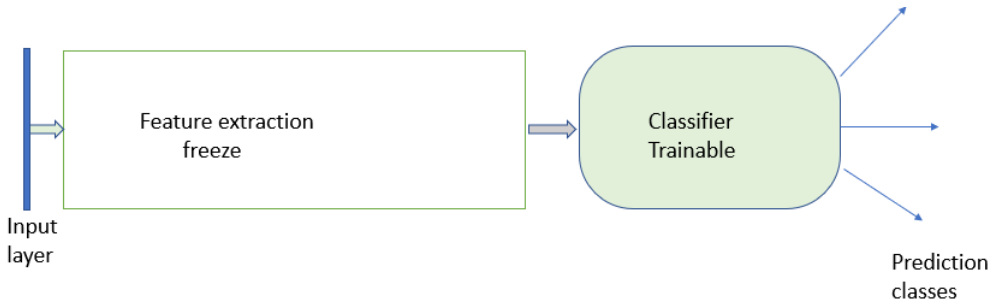


Figure 4.20: Freeze the segment of feature extraction of the CNN implemented, and a new classification segment is added

Therefore, as shown in figure 4.20, a new and specific to target dataset and task classifier segment is added to the CNN's frozen feature extractor. That classifier is trainable to the target dataset for the target task, while the feature extractor from the pretrained CNN is transferring the knowledge from the previous task that it did.

Thus, with the feature extraction technique, the frozen feature extractor and the new domain specific custom built trainable classifier are building the transfer learning CNN model.

Fine-Tuning:

For this technique, however, the last layers of the feature extraction segment of the pre-trained model are made to unfreeze and the learnt parameters contained in them are thereby trainable for the target dataset and thus allow better adaptation on the target dataset for the target task. The classifier is generally added new and trained on the target dataset together with the last layer trainable parameters [46].

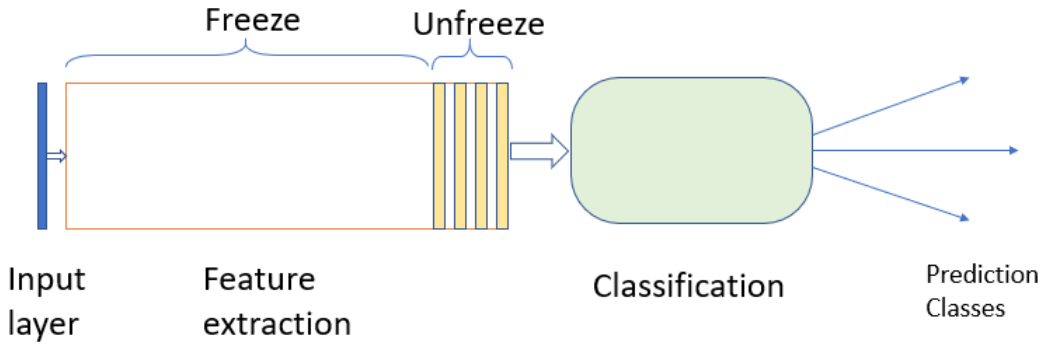


Figure 4.21: Fine-tuning through keeping in the last layers of the feature extractor trainable

For the fine-tuning approach of transfer learning, the feature extracor is not completely frozen but some of its last layers are kept trainable for the new target dataset and task, as shown in figure 4.21.

This is of a great significance that, the feature maps developed from the feature extractors, which are key factors for a successful model, are customized and learned too on the target dataset and task, along with their classifier. In this manner, the knowledge from a previous task is transferred with the feature extractor the CNN implemented, then that knowledge is customized through training on the target data and with the help of new and domain specific classifier the target task is done. The customization of the feature extraction layers can be done at any level as per required, for example, it is possible to unfreeze some of the last layers of the feature extractor or may be half of the layers of it, or even more of the layers.

Thus, with the fine-tuning technique, the partially unfrozen and trainable feature extractor and its new domain specific custom built trainable classifier are building the transfer learning CNN model.

4.3.2 Architectures of CNN models used

For transfer learning to be successful in such a target task of medical image processing and detecting Glaucoma, CNN models trained on a tremendous amount of images are required, so that the larger exposure to features of images might be effective.

Some of the CNN models that have been used in this work are-

- VGG
- ResNet
- DenseNet
- Inception

The CNNs mentioned have weights pre-trained on ImageNet and are available in keras as applications and have been loaded as such. ImageNet dataset has nearly 15 million high resolution images in 22,000 categories. ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where all of them performed outstanding, uses a subset of this gigantic dataset containing 1.2 million images in 1000 classes. A CNN trained on ImageNet means that it is extremely powerful and capable, since it is trained on 14 million images and thus has seen a tremendous number of features [44] [11] [47].

VGG

VGG [8] Convolutional Neural Network developed by Visual Geometry Group, University of Oxford have achebved admirable success with 16-19 layers of weights with small convolution filters of (3×3).

The figure 4.22 states the parameters number in millions and the deepest configuration has 19 weight layers in the column E. Convolutional layer notations are described as "conv(filter size)-(number of channels)". ReLU non-linearity is used here for all hidden layers [8].

In our proposed transfer learning model, we have used the 19 weight layer configuration of VGG commonly denoted as VGG19. The VGG19 architecture is depicted in figure 4.23.

ResNet

ResNet [14] or Residual Network won ILSVRC 2015, with outstanding performance carrying more than 150 layers which was way deeper than any model used before. ResNet makes such deeper models possible to train and perform outstandingly while neural networks who very deep used to have vanishing gradient problem and were difficult to work train well [14] [31].

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4.22: Number of parameters and ConvNet configuration [8]

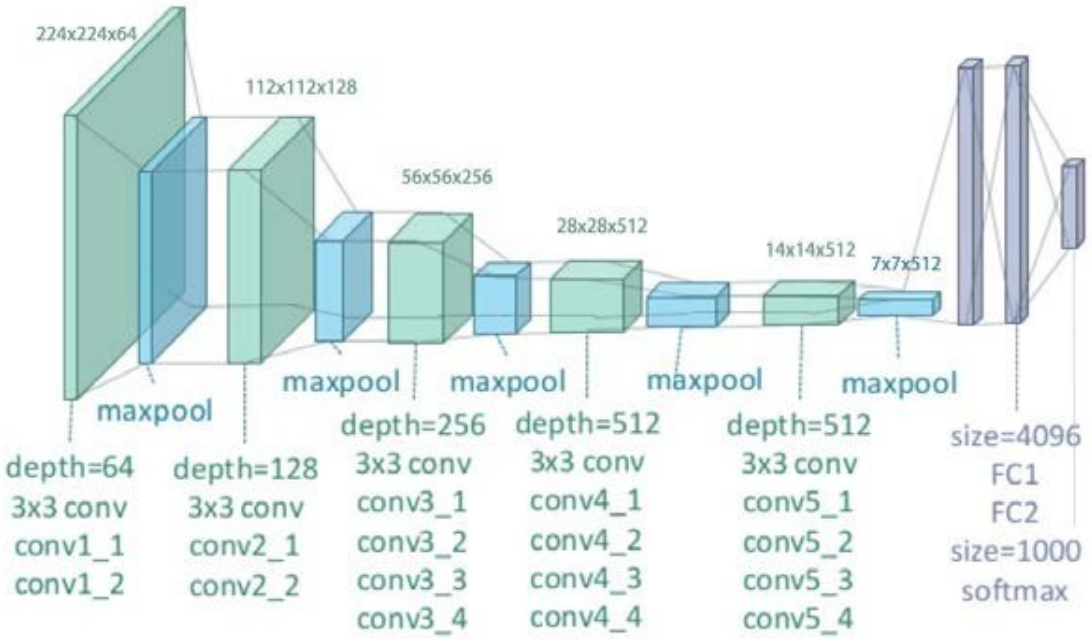


Figure 4.23: Architecture of VGG-19 [25]

In imagenet dataset, figure 4.24, for plain deep networks have higher error than shallow networks, while for ResNet deep network have lower error than shallow network.

ResNet avoids training error in deep networks by a concept of skip connections,

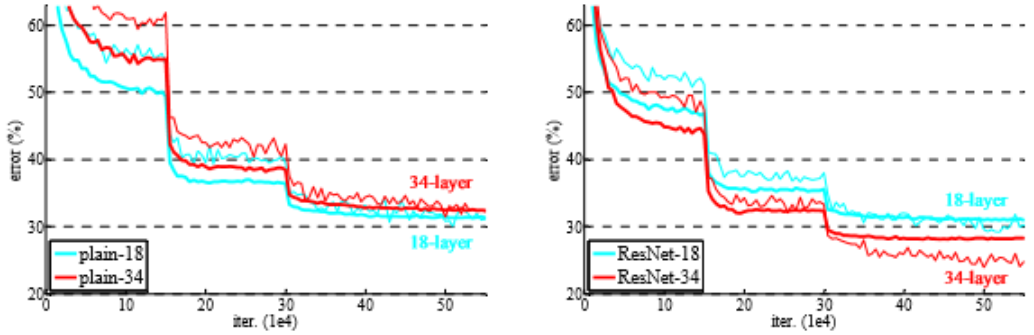


Figure 4.24: Training error for deep networks compared between plain and ResNet [14]

demonstrated in figure 4.25.

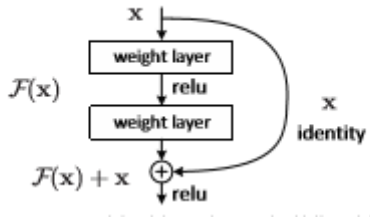


Figure 4.25: Residual learning: a building block [14]

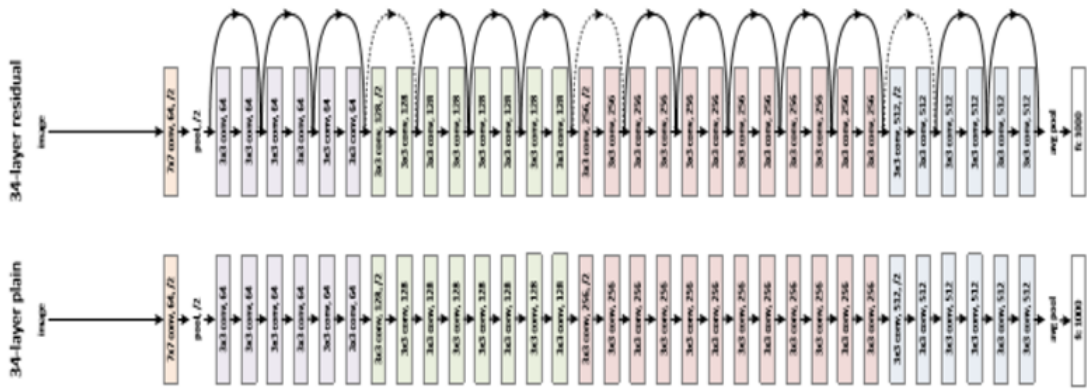


Figure 4.26: Resnet architecture is compared to a plain one with same number of layers [14]

A 34-layer ResNet architecture is demonstrated and compared with a 34-layer plain network, in figure 4.26, and the skip connections of the resnet is also showed.

ResNet architechtures for ImageNet is demonstrated in figure 4.27. The 50-layer architecture of ResNet, commonly denoted as ResNet50, is used in our proposed

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

Figure 4.27: Resnet networks described [14]

transffer learning model.

DenseNet

DenseNet or Dense Convolutional Network [17] architecture is made up of several dense blocks, where each layer is connected with all its preceding layers and take input from them and passes the output on to all the subsequent layers, within a dense block. [17].

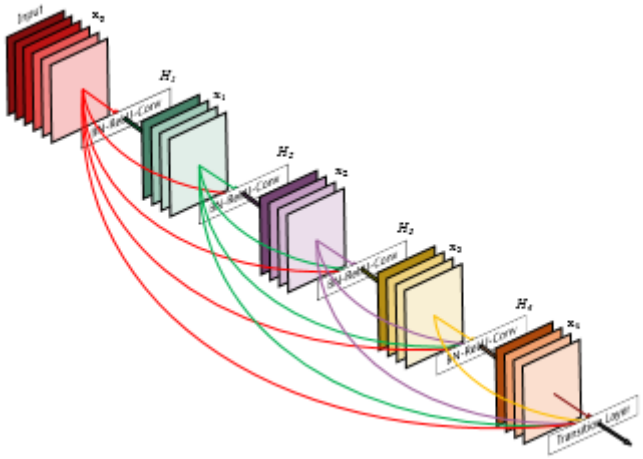


Figure 4.28: A five layer dense block [17]

Figure 4.28 demonstrating a dense block of five layers. Such dense blocks build a densenet architecture as shown in figure 4.29.

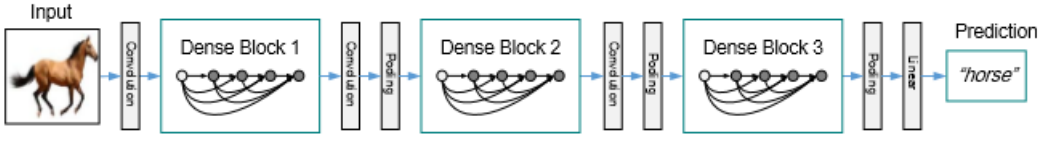


Figure 4.29: A densenet architecture containing three dense blocks [17]

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 4.30: DenseNet architectures for ImageNet [17]

InceptionV3

Inception [13] architecture is made up, through piling up several inception modules together, as a neural network of significant depth, where the inception modules provide the opportunity not to choose between filter sizes of convolution or even convolution and pooling themselves but to operate them all in parallel and concatenate their outputs into a single output vector for being input to the module afterwards.

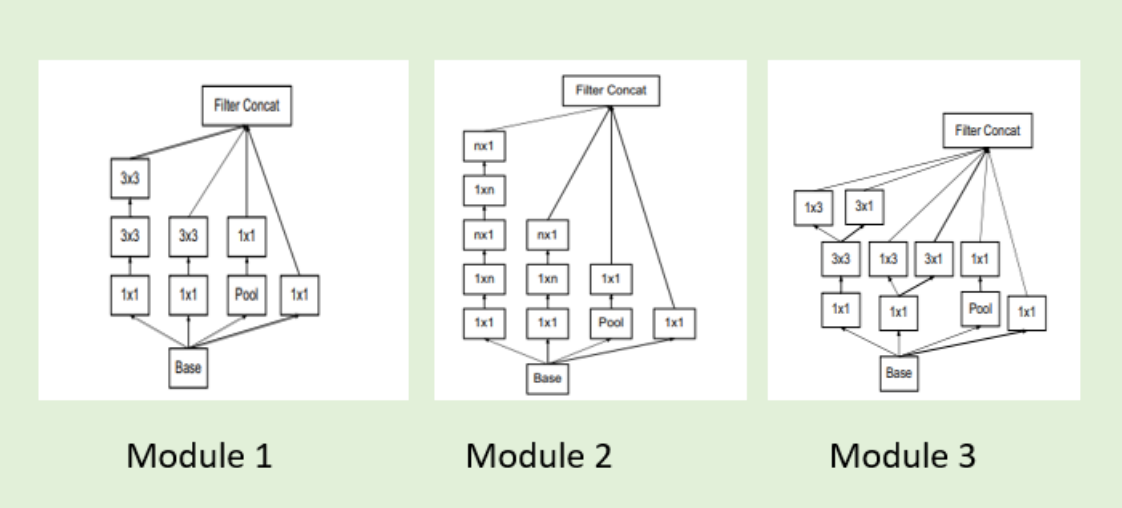


Figure 4.31: The three inception modules [15]

InceptionV3, a new variant of inception architecture, is used in our transfer learning model proposed [15]. The outline of the InceptionV3 network architecture, table 4.1. The output size of each module is the input size of the next one.

The module in figure 4.32 helps reducing grid size of feature maps [15].

Type	patch size/stride or remarks	input size
conv	3 ×3/2	299 ×299 × 3
conv	3 ×3/1	149 ×149 × 32
conv padded	3 ×3/1	147 ×147 × 32
pool	3 ×3/2	147 ×147 × 64
conv	3 ×3/1	73 ×73 × 64
conv	3 ×3/2	71 ×71 × 80
conv	3 ×3/1	35 ×35 × 192
3 ×*Inception*	module 1	35 ×35 × 288
5 ×*Inception*	module 2	17 ×17 × 768
2 ×*Inception*	module 3	8 ×8 × 1280
pool	8 ×8	8 ×8 × 2048
linear	logits	1 ×1 × 2048
softmax	classifier	1 ×1 × 1000

Table 4.1: Outline of the InceptionV3 architecture [15]

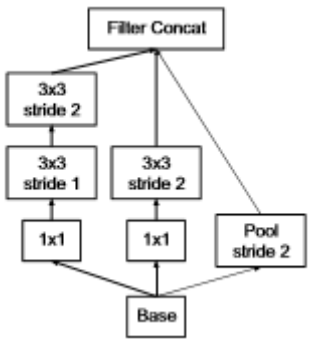


Figure 4.32: Module to reduce filter grid size [15]

Chapter 5

Methodology

A Transfer Learning approach is proposed in this research. The size and characteristics of the data set create an ideal ground to implement a transfer learning methodology so that a pre-trained CNN with all its weights can be used for building up a new transfer learning model dedicated to the task of detecting Glaucoma with a satisfactory level of accuracy.

Since, the data set is not very small either, the Feature Extraction approach is not used rather the Fine-tuning approach is used.

5.1 The libraries, tools and software used

Python: Python is a high level programming language that is capable of object-oriented programming and structured programming. The major strength of python is its large standard library that provides tools for a vast variety of tasks, like Image processing and Machine learning etc.

Ipython [3] interactive computing has been used, which provides a rich toolkit for interactive usage of python, having components:

- A pyhton shell
- A Jupyter kernel for Jupyter notebook usaged

Jupyter Notebooks: Jupyter notebooks is used, that is an open source web application. It facilitates live code, equations, visualizations and narrative text [45].

TensorFlow: TensorFlow [10] open source machine learning platform offering a friendly ecosystem of tools, libraries and resources.

Keras: Keras [9] is a high-level neural networks API, with TensorFlow in its backend.

Matplotlib: Matplotlib [2] is used in our research, which is a inclusive library for interactive visualizations in Python.

5.2 Implementation of Transfer Learning

The Fine-tuning approach of Transfer learning is implemented by retraining the whole previously trained CNN along with the customised classifier block added that builds up the transfer learning model. We haven't kept any of the layers from the pre-trained CNN model in freeze rather made all of the layers with their parameters to be trainable, utilizing the pre-trained weights and biases as an optimized initialization for the intended task of detecting Glaucoma. The new model is organized linearly with keras using the Sequential model type.

We propose four transfer learning models with the four pre-trained CNNs, and we compare them in terms of methodology and outcome or result.

5.2.1 Dataset reorganization

The fundus images along with all their associated labels are further distributed randomly into three distinct sets, as such:

- Train set
- Validation set
- Test set

The split of labeled fundus image data into these three sets is done to make the effective use of the data.

Set of data	Suspecious Glaucoma		Non Glaucoma	
	fundus images	percentage	fundus images	percentage
Train set	1199	70%	2201	70%
Validation set	256	15%	471	15%
Test set	256	15%	471	15%

Table 5.1: Distribution of labeled fundus images into train, validation and test sets

Table 5.1 demonstrates the number of fundus images along with their labels from each class and the corresponding percentage distributed into the train set, validation set and the test set. The percentage of image data and their labels split into train, validation and test sets are 70%, 15% and 15% respectively.

5.2.2 Architecture of the proposed model

The Transfer Learning models proposed are having an architecture shown in Fig. 5.1

The previously trained CNN models are imported and loaded from keras as applications in a form that their classifier blocks get dropped off. Thus the pre-trained

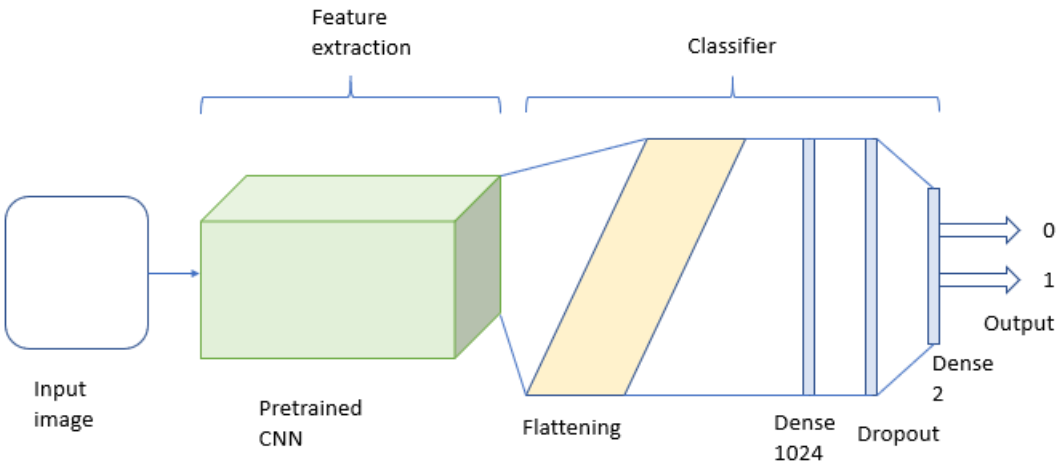


Figure 5.1: The Architecture of the Transfer learning model proposed

CNN loaded contain only the feature extractor where all the knowledge from ImageNet training is reside. The classifier block of a previously trained CNN model does not help in our target dataset and task.

Thereby, the two segments of our proposed Transfer learning models are as such:

- Feature extraction: The feature extractor segment of our proposed model is transferred from the pretrained CNN model. The pretrained CNN model is loaded only with its feature extractor carrying all its learned features, which is used as the feature extractor in our model.

- Classifier: The classifier of our proposed transfer learning model is created distinctly for our target dataset and task.

The model architecture is dedicated to the task of classifying fundus images for suspicious Glaucoma. Once the training is done with all parameters set, a model architecture summary is shown as Fig. 5.2

5.2.3 Previously trained CNN

From the classifier excluded pre-trained CNN, the last layer is taken, so that the full feature extractor can be transferred into our transfer learning model, and termed as $transfer\_layer$. In order to create a transfer learning model, a model is created defining the input of the loaded pre-trained CNN as its input and the output from the $transfer\_layer$ as its output. Thus, a model is built to operate with. Using

$$.add()$$

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model (Model)                (None, 7, 7, 512)         20024384
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              25691136
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 2050
=================================================================
Total params: 45,717,570
Trainable params: 45,717,570
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.2: Summary of the Model(with VGG19) architecture

method further layers are added to complete the desired transfer learning model. The "*transfer_layer*"s of the four previously trained CNN models, that are being used:

- VGG19 : The last layer is *block*5_*pool* that is takes as transfer layer, as marked Red in Fig. 5.3

```
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
_________________________________________________________________
```

```
transfer_layer = model.get_layer('block5_pool')
conv_model = Model(inputs = model.input, outputs = transfer_layer.output)
```

Figure 5.3: Taking the transfer layer for VGG19

- ResNet50 : The last layer taken is *conv*5_*block*3_*add* as transfer layer, getting the whole ResNet50 pre-trained CNN. Marked Red in fig 5.4

- DenseNet121: The last layer taken is *bn* excluding the activation layer, with which the full DenseNet121 pre-trained CNN is taken. Marked Red in fig 5.5

- InceptionV3: layer *activation*_93 is used as transfer layer to get the full inceptionV3. Marked Red in fig 5.6

```
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0][0]
==================================================================================================
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
```

```
In [9]: transfer_layer = model.get_layer('conv5_block3_add')
        conv_model = Model(inputs = model.input, outputs = transfer_layer.output)
```

Figure 5.4: Taking the transfer layer for ResNet50

```
bn (BatchNormalization)         (None, 7, 7, 1024)   4096        conv5_block16_concat[0][0]
relu (Activation)               (None, 7, 7, 1024)   0           bn[0][0]
==================================================================================================

Total params: 7,037,504
Trainable params: 6,953,856
Non-trainable params: 83,648
```

```
transfer_layer = model.get_layer('bn')
conv_model = Model(inputs = model.input, outputs = transfer_layer.output)
```

Figure 5.5: Taking the transfer layer for DenseNet121

```
activation_93 (Activation)      (None, 5, 5, 192)    0           batch_normalization_93[0][0]
mixed10 (Concatenate)           (None, 5, 5, 2048)   0           activation_85[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_1[0][0]
                                                                 activation_93[0][0]
==================================================================================================
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
```

```
]: transfer_layer = model.get_layer('activation_93')
   conv_model = Model(inputs = model.input, outputs = transfer_layer.output)
```

Figure 5.6: Taking the transfer layer for InceptionV3

5.2.4 Classifier block

A resolute classifier block is added to the previously trained CNN with configurations as such:

- A Flatten layer in order to flatten the output from the Convolutional layer.
- A Dense layer or a fully connected layer of 1024 units having Sigmoid activation function. The Sigmoid activation is used to help the probability distribution to be more accurate and polarized.
- A dropout layer to minimize over fitting. The dropout layer helps by dropping a portion of connections and thus reduce over fitting to some extent. Dropouts set as per Table 5.2

Pre-trained CNN	Dropout
VGG19	0.25
ResNet50	0.5
DenseNet121	0.5
InceptionV3	0.3

Table 5.2: Dropout used for the corresponding CNN

- Another Dense layer or a fully connected layer is added as the final classification layer with units of 2 as per the number of classes. The activation function used for this layer is Softmax activation function.

5.2.5 Compilation

The model built with fine-tuning approach then needs to be complied. In order to compile the model the Loss function and the Optimizer is defined:

- Loss function: Throughout the learning process the loss function calculates the error of prediction from the label. The data set in its two classes are mutually exclusive. The loss function used is *categorical_crossentropy*.

- Optimizer: The optimizer takes the loss into account and corrects it. Thus the weights of the transfer learning model can be further adjusted. Optimizer 'Adam' is used. The learning rates used as per Table 5.3 for the four different CNN containing models.

Pre-trained CNN	learning rate
VGG19	1e-5
ResNet50	0.0001
DenseNet121	0.0001
InceptionV3	0.0001

Table 5.3: Learning rate with corresponding CNN

5.2.6 Data preprocessing

Since we are working with image data in our target data set, the data preprocessing is doen with *ImageDataGenerator*() class, that provides real time data augmentation.

The images here in the target data set are colored. All the colors in an image are made by colored pixels, each made up with 3 channels of colors Red, Green and Blue or known as 'RGB' in short. Each color channel, for 1 byte made up of 8 bits, is valued from 00000000 to 11111111 in binary or 0 to 255 in decimal. The maximum binary number for a byte or 8 bits, 11111111 translates 255 into decimal. Therefore, a color pixel is defined with 3 channels of colors 'RGB' each having values in a range between 0 to 255.

The images are rescaled so as to get the pixel values in the range between 0 to 1 instead of 0 to 255, through dividing all the pixels by 255.

5.2.7 Training the model

After being compiled the transfer learning model is ready to be trained on the intended data set. The train set and validation set were used for training the model as well as validating it simultaneously.

The training of the model is done with:

- Training data: The train set of the dataset, which contain 70% of the image data, is assigned.
- Epoch: The number of epochs used to train and validate the models, is given in the Table. 5.4

Pre-trained CNN used in the model	number of epochs
VGG19	40
ResNet50	40
DenseNet121	40
InceptionV3	30

Table 5.4: Number of Epochs with corresponding CNNs used in the model

- Steps per epoch: Number of steps or batches of data in each completed epoch[9]. Thus 36.35 is used as the steps per epoch, calculated as number of images in data divided by the batch size.
- Validation data: The validation set extracted from the data set, which carries 15% of the image data, is mounted as validation data here for validating the training or learning process.
- Validation steps: It is calculated as same as the steps per epoch. Thus it is set as 36.35 matching the steps per epoch value.

5.2.8 Saving models

When the training is complete, the model with all its trained parameters(adjusted weights and biases), is then saved as a

$$.h5$$

file. The saved models in a form of $.h5$ file can be used for further testing and usage.

Chapter 6

Result and analysis

6.1 Train and validation: Learning and Gaining accuracy

Accuracy of prediction is core of the outcome expected for deep learning. After performing all the epochs of training process the model predicts the training set with an optimal accuracy and a loss calculated, comparing the predictions from the model with the true labels provided with the data. The training set predictions are simultaneously validated with the validation set, providing a validation accuracy and validation loss.

6.1.1 VGG19

The model built with VGG19 performs smoother to gain accuracy and reducing loss during training process. The gradual decline of loss over epochs is showed in the figure 6.1

Figure 6.1: Loss vs Number of Epochs

Figure 6.1 is showing that the validation loss is following the train loss mostly in the same path. Although the train loss is a slightly lower than the validation loss,

indicating a small and an almost negligible over-fitting.

On the other hand training accuracy is increasing over number of epochs and the validation accuracy is following, shown in fig. 6.2.

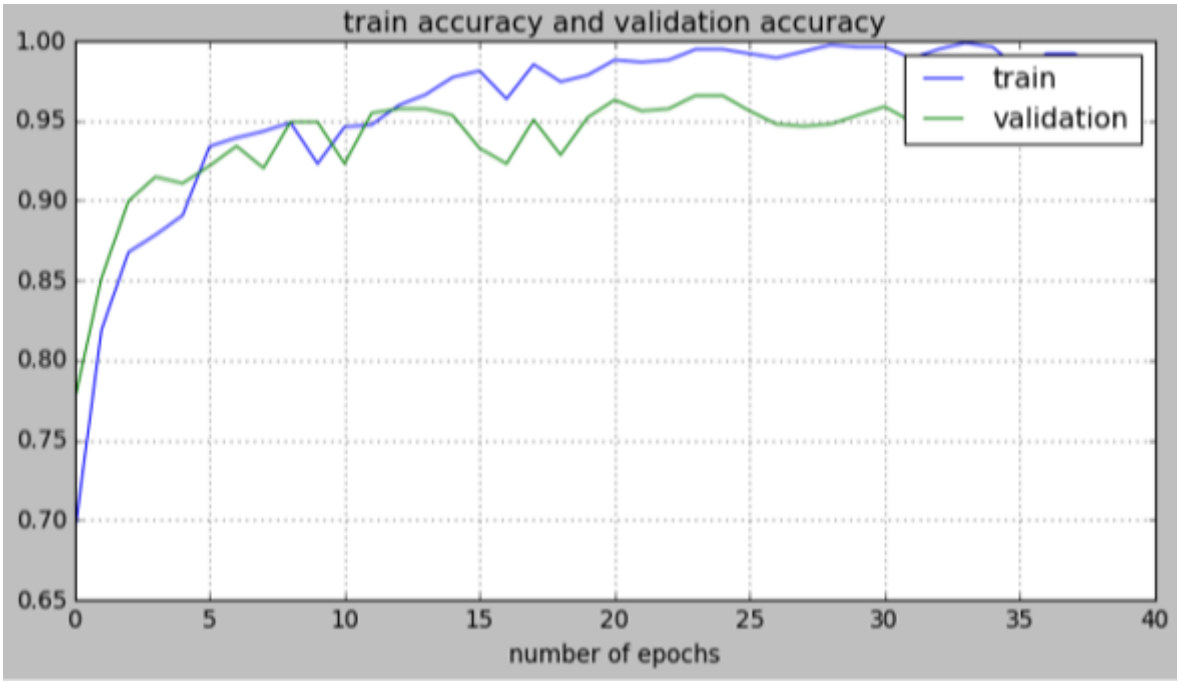


Figure 6.2: Accuracy vs Number of Epochs

The graph of accuracy is reverse of the loss, shown in figure 6.3.

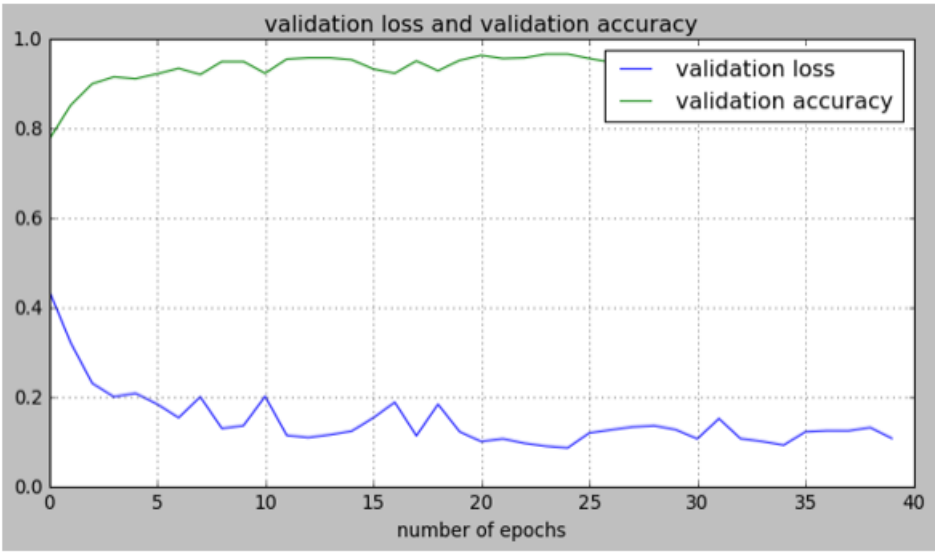


Figure 6.3: Validation Loss and validation accuracy

The Validation loss gets close to 0.0 while the validation accuracy gets close to 1.0 On completion of 40 epochs the validation loss is 0.1080 and validation accuracy

is 0.9574, while initially the validation loss was 0.4384 and the validation accuracy was 0.7744, showing in fig 6.4 and fig 6.5

```
Train for 36.35 steps, validate for 36.35 steps
Epoch 1/40
37/36 [==============================] - 1319s 36s/step - loss: 0.5812 - categorical_accuracy: 0.6905 - val_loss: 0.4384 - va
l_categorical_accuracy: 0.7744
Epoch 2/40
37/36 [==============================] - 1309s 35s/step - loss: 0.3978 - categorical_accuracy: 0.8189 - val_loss: 0.3217 - va
l_categorical_accuracy: 0.8514
Epoch 3/40
37/36 [==============================] - 1364s 37s/step - loss: 0.3074 - categorical_accuracy: 0.8676 - val_loss: 0.2308 - va
l_categorical_accuracy: 0.8996
```

Figure 6.4: The validation loss and validation accuracy at initial epoch

```
Epoch 39/40
37/36 [==============================] - 1457s 39s/step - loss: 0.0422 - categorical_accuracy: 0.9851 - val_loss: 0.1318 - va
l_categorical_accuracy: 0.9532
Epoch 40/40
37/36 [==============================] - 1433s 39s/step - loss: 0.0344 - categorical_accuracy: 0.9892 - val_loss: 0.1080 - va
l_categorical_accuracy: 0.9574
```

Figure 6.5: The validation loss and validation accuracy at final epoch

6.1.2 ResNet50

Using ResNet50 pre-trained CNN in the transfer learning model, the validation loss initially showed a no reduction compared to the training loss. Which indeed caused a no initial increase in validation accuracy compared to the training accuracy. However, after nearly half of the total epochs the validation loss started to decrease rapidly and the validation accuracy did rise close to the training accuracy to match the corresponding validation loss. Demonstrated in fig 6.6 and fig 6.7.

Figure 6.6: Loss vs Number of Epochs (using ResNet50)

Fig 6.7 shows the validation accuracy was a consistent value of 0.65 or 65% during the initial 17 epochs, and afterwards it increased up to more than 0.95 or 95% which is very close to the training accuracy off 0.98 or 98%.

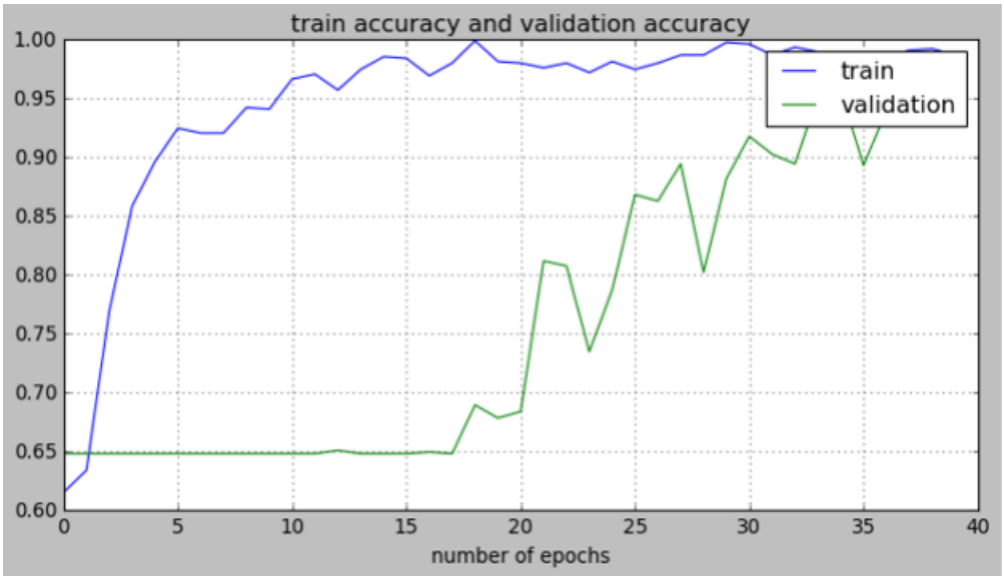


Figure 6.7: Accuracy vs Number of Epochs (using ResNet50)

The validation accuracy is represented comparing the validation loss in fig 6.8.

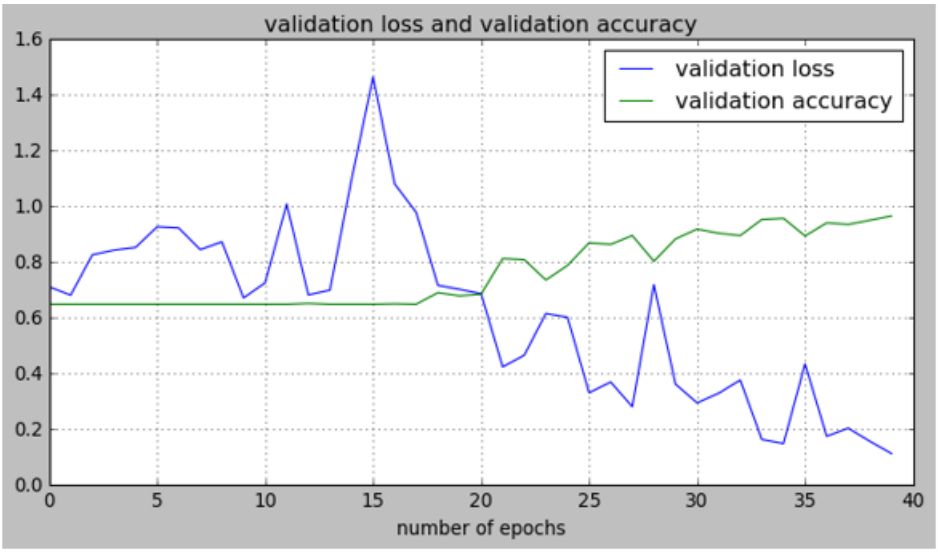


Figure 6.8: Validation loss and Validation accuracy

6.1.3 DenseNet121

The DenseNet121 model is a very dense architecture to work with and thus the model built with DenseNet121 did not show a smoother graph. The reduction of validation loss compared to reduction of training loss was consistent, however, it did show steep increase twice throughout the training process with our data set around epochs 4 to 5 and 19 to 20. After epoch 20 the curve remained nearly stable and the validation loss finally reduced down to 0.2 at the final epoch, shown in fig 6.9

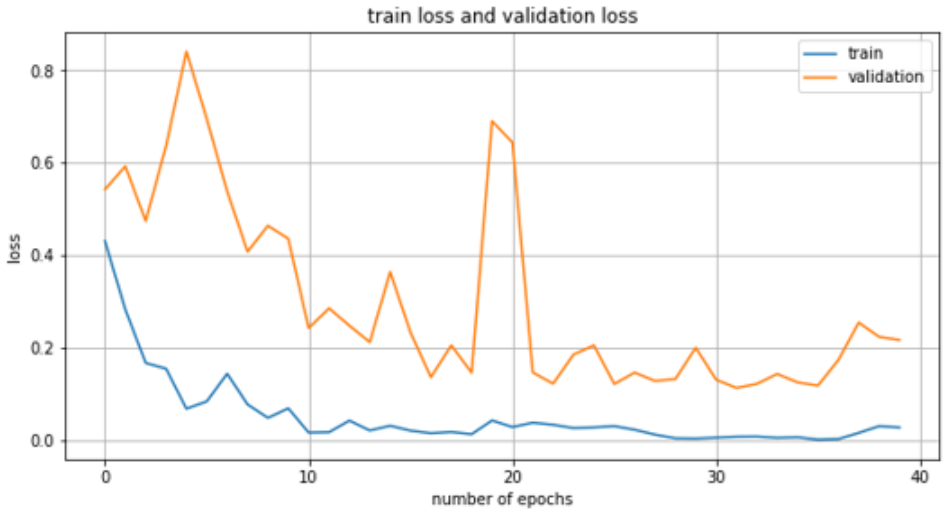


Figure 6.9: Loss vs Number of Epochs (using densenet121)

The accuracy for both the training and validation, shown in fig 6.10, thus showed outcome corresponding to the losses.

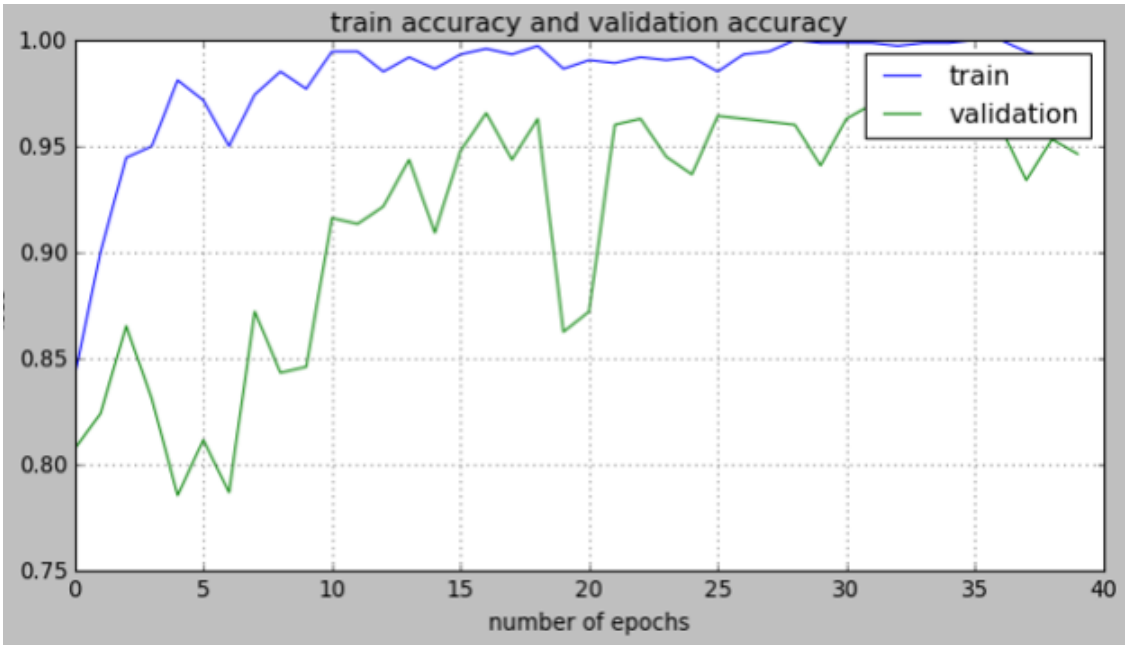


Figure 6.10: Accuracy vs Number of Epochs

Validation accuracy is compared to the validation loss in fig 6.11.

The DenseNet121 did finally acheived a validation accuracy of 0.95 or 95%.

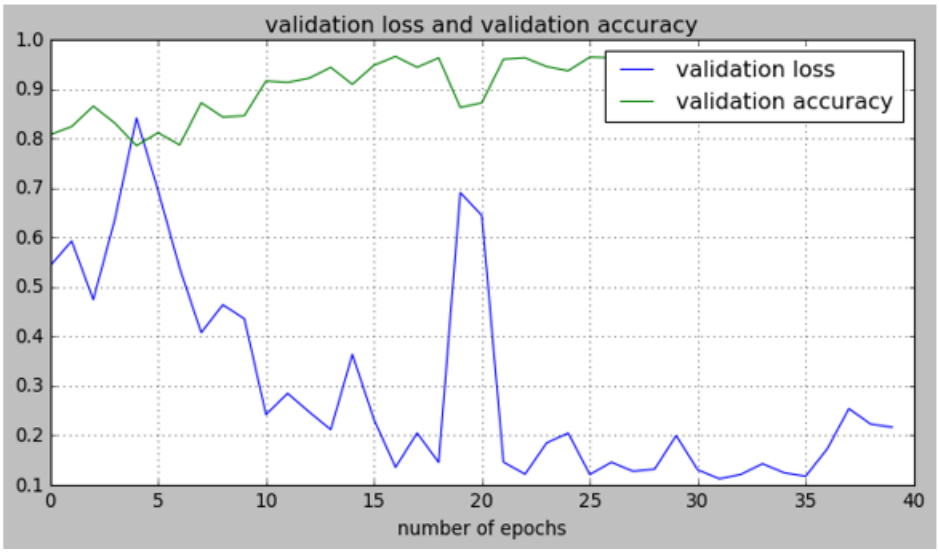


Figure 6.11: Validation Loss and validation accuracy

6.1.4 InceptionV3

The model built using InceptionV3 is a well performing model. It showed a steeper drop of loss both of training and validation within initial epochs and mostly maintained a consistent value of loss for the rest of the epochs, demonstrated in fig 6.12

Figure 6.12: Loss vs Number of Epochs(using InceptionV3)

The accuracy for both the training and validation did show the outcome corresponding the losses, in fig 6.13

The Validation loss and accuracy are reverse of each other showed in fig 6.14

The model finally achieved a validation accuracy of 0.92 or 92%.

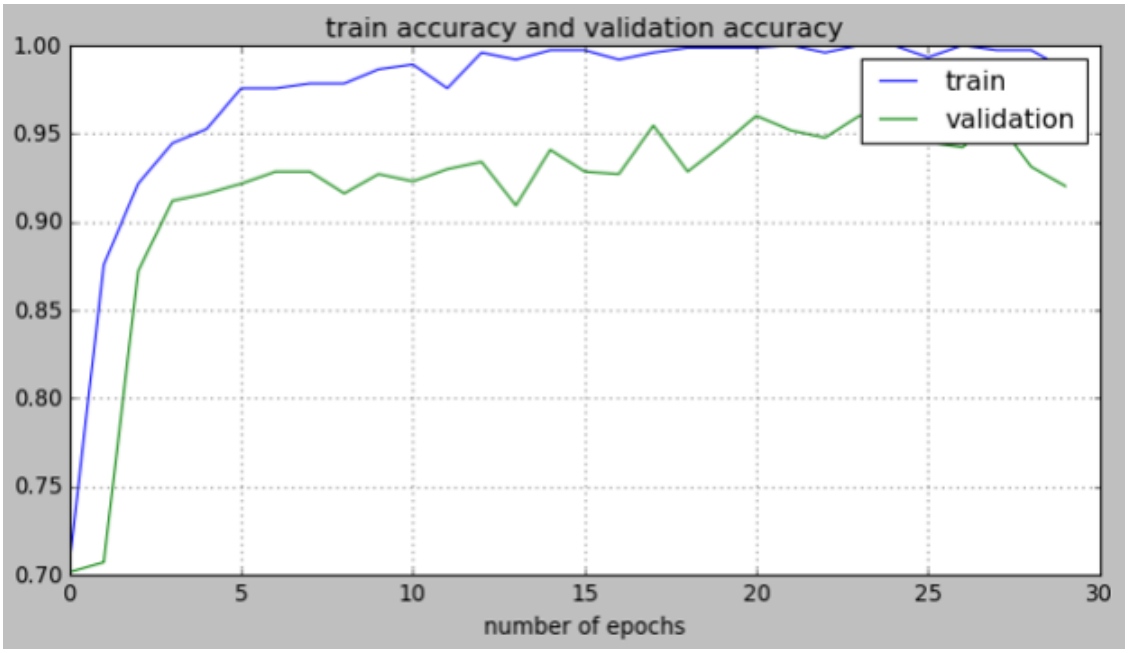


Figure 6.13: Accuracy vs Number of Epochs(using InceptionV3)

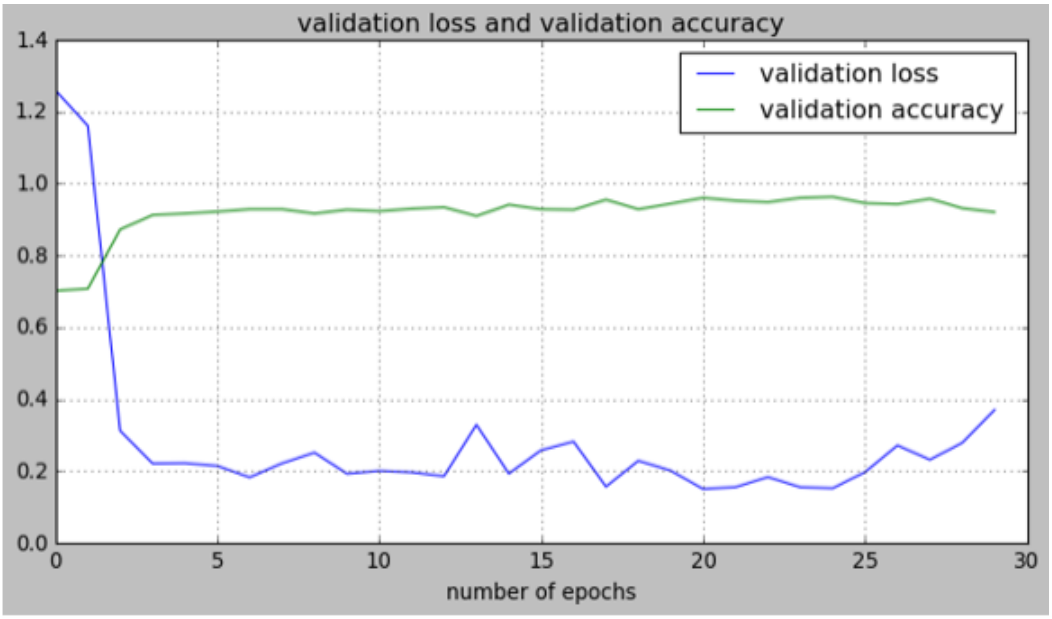


Figure 6.14: Validation Loss and validation accuracy compared ober number of epochs

6.2 Test

The transfer learning models built with pre-trained CNNs are retrained and validated on the target data set with the train set and validation set. All the four models that are saved as .$h5$ models are further subject to test their performances. Testing of the models are done with the test set extracted from the LAG data set, and the test set is completely distinct from both the train set and validation set.

The .$h5$ models are loaded in a notebook using *load_model*() and the test set is

mounted, being processed with $ImgaeDataGenerator()$.

The models are evaluated on the test set.

- VGG19: For the model containing VGG19 the test set evaluation shows accuracy of 0.945 or 94.5%, as in fig 6.15

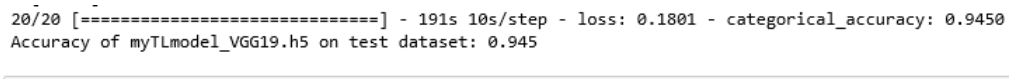


Figure 6.15: Test accuracy and loss for VGG19 built model

- ResNet50: For the model containing ResNet50 the test set evaluation shows an accuracy of 0.96 or 96.0% and a loss of 0.08, as in fig 6.16

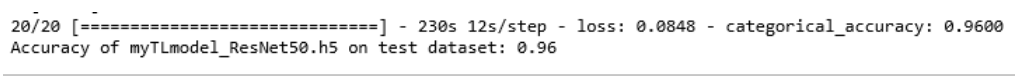


Figure 6.16: Test accuracy and loss for ResNet50 built model

- DenseNet121: For the model containing DenseNet121 the test set evaluation shows an accuracy of 0.925 or 92.5% and a loss of 0.3862, as in fig 6.17

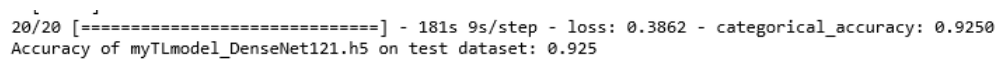


Figure 6.17: Test accuracy and loss for DenseNet121 built model

- InceptionV3: For the model containing DenseNet121 the test set evaluation shows an accuracy of 0.9175 or 91.75% and a loss of 0.3784, as in fig 6.18

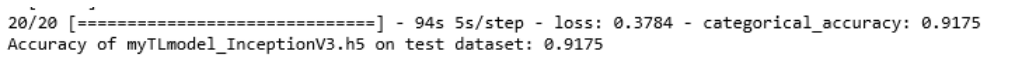


Figure 6.18: Test accuracy and loss for InceptionV3 built model

The accuracy of the models on test set profoundly matches that of the corresponding validation accuracy, demonstrated in the table 6.1. Thus signifying that the models are built and trained aptly on the target data set, and working as per expectation to the target task of detecting Glaucoma from the Fundus images.

Used pre-trained CNN	Test accuracy	Validation accuracy
VGG19	0.945 or 94.5%	0.9574 or 95.74%
ResNet50	0.96 or 96%	0.98 or 98.0%
DenseNet121	0.925 or 92.5%	0.95 or 95.0%
InceptionV3	0.9175 or 91.75%	0.92 or 92%

Table 6.1: Comparison on Test accuracy and validation accuracy

6.3 Prediction

Since the models are tested on the test set, they are subject to show some predictions for Glaucoma on the fundus images belonging to the test set.

6.3.1 VGG19

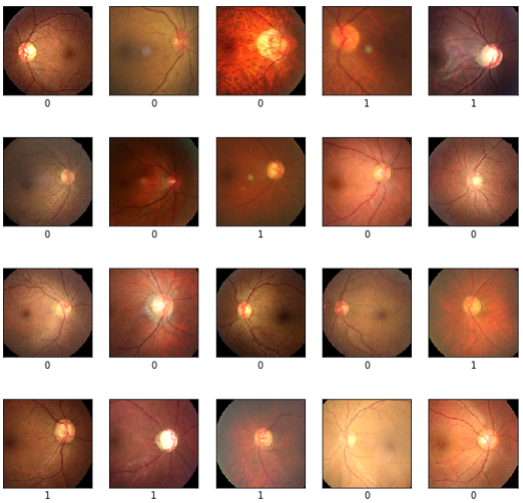


Figure 6.19: Predictions from transfer learning model with VGG19

Figure 6.19 is showing the predictions for random 20 images from the test set, made by the saved model of *.h*5 file developed with transfer learning that carries VGG19 and the figure 6.20 is showing the labels for the images. The label of an image is denoted with the number in the right-hand side inside the parentheses.

6.3.2 ResNet50

Figure 6.21 is showing the predictions for random 20 images from the test set, made by the saved model of *.h*5 file developed with transfer learning that carries ResNet50 and the figure 6.22 is showing the labels for the images.The label of an image is denoted with the number in the right-hand side inside the parentheses.

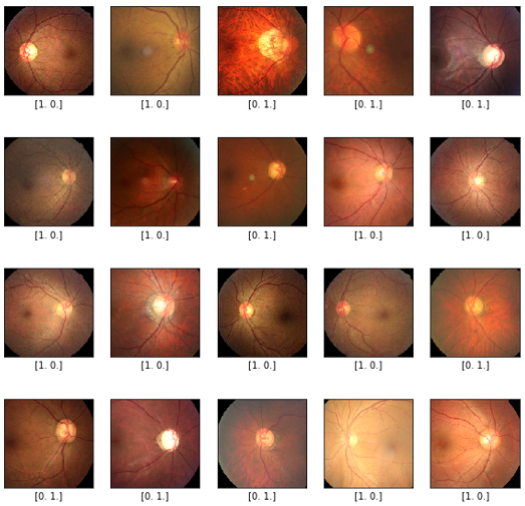


Figure 6.20: Corresponding labels of the images in figure 6.19

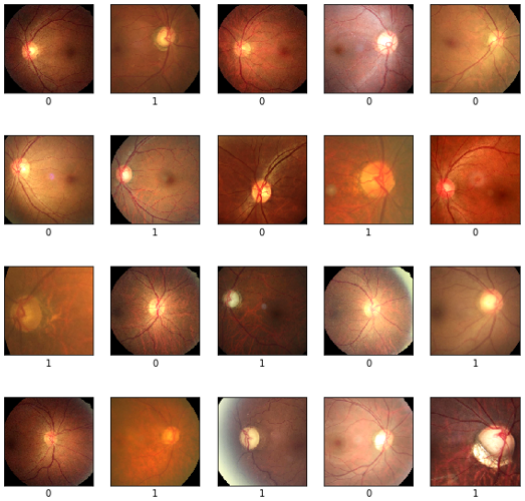


Figure 6.21: Predictions from transfer learning model with ResNet50

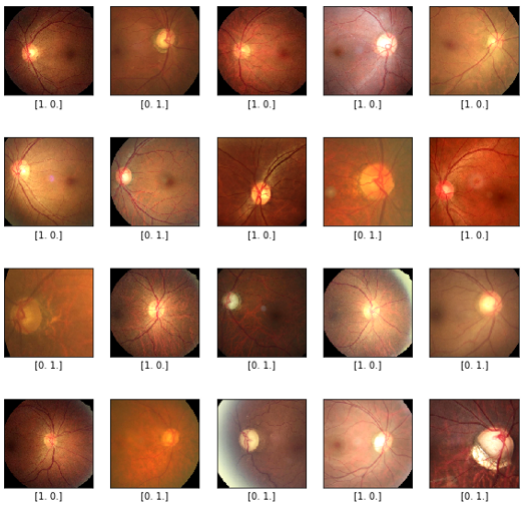


Figure 6.22: Corresponding labels of the images in figure 6.21

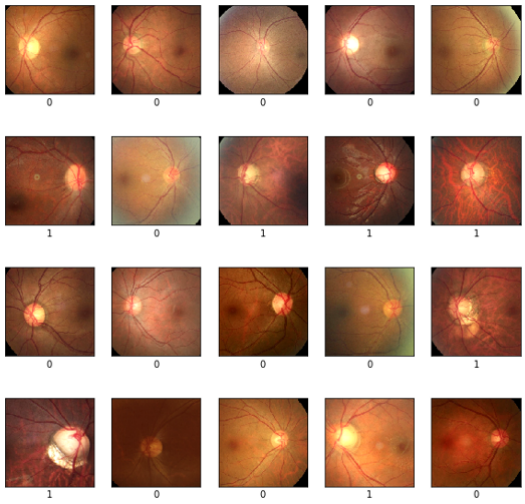


Figure 6.23: Predictions from transfer learning model with DenseNet121

6.3.3 DenseNet121

Figure 6.23 is showing the prediction for random 20 images from the test set, made by the saved model of .$h5$ file developed with transfer learning that carries DenseNet121 and the figure 6.24 is showing the labels for the images. The label of an image is denoted with the number in the right-hand side inside the parentheses.

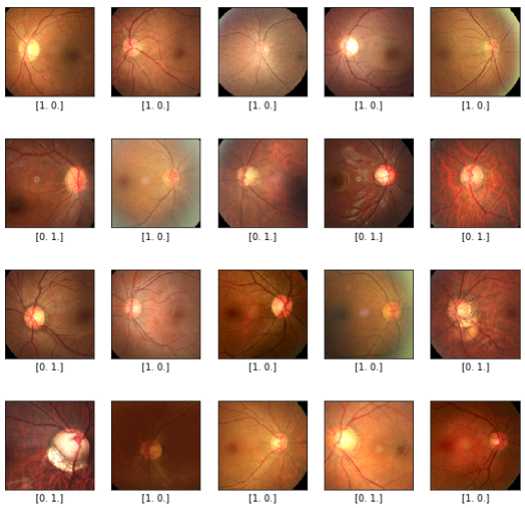


Figure 6.24: Corresponding labels of the images in figure 6.23

6.3.4 InceptionV3

Figure 6.25 is showing the predictions for random 20 images from the test set made by the saved model of .$h5$ file developed with transfer learning that carries InceptionV3 and the figure 6.26 is showing the labels for the images. The label of an image is denoted with the number in the right-hand side inside the parentheses.

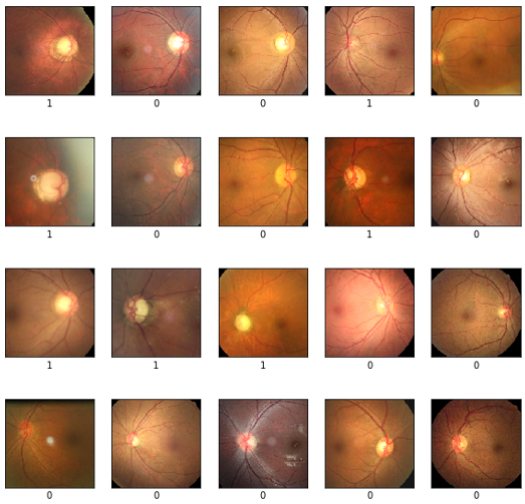


Figure 6.25: Predictions from transfer learning model with InceptionV3

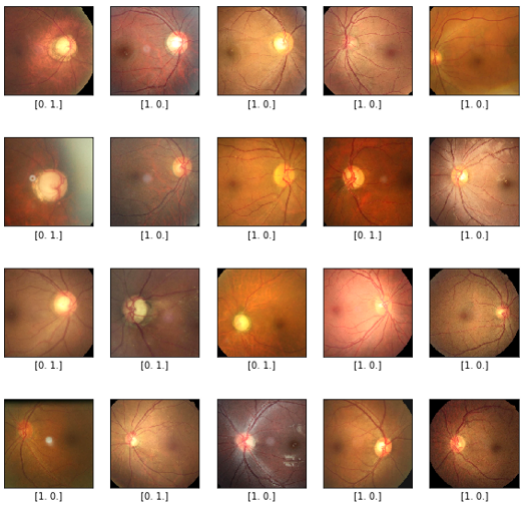


Figure 6.26: Corresponding labels of the images in figure 6.25

6.4 Outcome Evaluation

We evaluated our models with respect to L. Li *etal.*(2019)[33], Guangzhou *etal.* (2019)[26] and Perdomo *etal.*(2018)[21].

L. Li *etal.*(2019) [33] with their AG-CNN model achieved an accuracy of 95.3% on their test set of the LAG database. Guangzhou *etal.* (2019)[26] had accuracy evaluated for the CNNs were 0.940 or 94.0% for fundus images used in grayscale, where their dataset was very small. Perdomo *etal.*(2018)[21] acheived an accuracy of 89.4% using multi stage deep learning model. Having a small dataset may lead to biased prediction such as over-fitting or under-fitting.

Our best performing transfer learning model is the Resnet50 built model which is providing 96.5% accuracy on our test set of the LAG database, which is higher than that achieved by [33], [26], [21], and the VGG19 built model of ours is providing an accuracy of 94.75% on the test set of LAG database. The other two transfer

learning models built with DenseNet121 and InceptionV3 is also comparable with AG-CNN, achieving accuracy of 92.5% and 91.75% respectively. It is to mention that, LAG database we used contain a comparatively large number of fundus image data of 3400 images in two classes for train set, and 727 images in two classes for validation set. Since, all of our proposed models having accuracy higher than 90%, therefore, the dataset is well distributed and the predictions are fair. Thus, our proposed transfer learning methodology, with CNNs VGG19, ResNet50, DenseNet121 and InceptionV3, has achieved a similar level of accuracy as [33], [26] and [21] has achieved for detecting Glaucoma from fundus images. The comparison is demonstrated in table 6.2

methodology	accuracy
L. Li *etal.*(2019)[33]	95.3%‘
Guangzhou *etal.* (2019)[26]	94.0%
Perdomo *etal.*(2018)[21]	89.4%
Proposed model with ResNet50	**96.0%**

Table 6.2: Our proposed model of highest accuracy is compared

Chapter 7

Future scope and Conclusion

7.1 Future Scope

In future, this transfer learning research can be carried on from where we have left and try to improve the accuracy for all the models.

Since the ResNet50 built transfer learning model has given the best result here, this model may be used in a hardware system, which might help assisting diagnosis of Glaucoma at an early stage before the condition get acute.

Nowadays, there are many IT companies, corporate offices or TV channels where people have to work with computers all day long. Sometimes, people suffer from different eye diseases and pain due to keeping eyes on the computer screens for a long time. As we know, glaucoma does not show its symptoms at an early stage, a regular eye diagnosis is important for everyone. In this case, any of the glaucoma detector models can be used in these offices so that the employees can checkup their eyes whenever they need.

7.2 Conclusion

In this paper, we worked with four different Convoluted Neural Network models. We did a comparative study using these models. All the four models developed, delivered spectacular performances on detecting Glaucoma on a fundus images from test set, that they never saw before. Among all the four models, ResNet50 has shown the highest accuracy of 96.75% on the test set images. In conclusion, it can be said that this comparative study may play a significant role for developing any new model to diagnose Glaucoma, and it can be a gateway for starting a new approach or a research to carry on.

Bibliography

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] J. D. Hunter, "Matplotlib: A 2d graphics environment", *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[3] F. Pérez and B. E. Granger, "IPython: A system for interactive scientific computing", *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21–29, May 2007, ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53. [Online]. Available: https://ipython.org.

[4] J. Nayak, R. Acharya, P. S. Bhat, N. Shetty, and T.-C. Lim, "Automated diagnosis of glaucoma using digital fundus images", *Journal of medical systems*, vol. 33, no. 5, p. 337, 2009.

[5] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[6] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.

[9] F. Chollet *et al.*, *Keras*, 2015.

[10] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge", *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[12] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural networks*, vol. 61, pp. 85–117, 2015.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[16] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification", in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, 2017, pp. 721–724.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[18] T. Khalil, M. U. Akram, S. Khalid, and A. Jameel, "An overview of automated glaucoma detection", *2017 Computing Conference*, pp. 620–632, 2017.

[19] *Glaucoma*, Available at https://www.mayoclinic.org/diseases-conditions/glaucoma/symptoms-causes/syc-20372839, Nov. 2018.

[20] L. Hulstaert, *Transfer learning: Leverage insights from big data*, Available at https://www.datacamp.com/community/tutorials/transfer-learning, 2018.

[21] O. Perdomo, V. Andrearczyk, F. Meriaudeau, H. Müller, and F. A. González, "Glaucoma diagnosis from eye fundus images based on deep morphometric feature estimation", in *Computational pathology and ophthalmic medical image analysis*, Springer, 2018, pp. 319–327.

[22] S. Saha, *A comprehensive guide to convolutional neural networks — the eli5 way*, Available at https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2018.

[23] D. (Sarkar, *A comprehensive hands-on guide to transfer learning with real-world applications in deep learning*, Available at https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a, 2018.

[24] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology", *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[25] Y. Zheng, C. Yang, and A. Merkulov, "Breast cancer screening using convolutional neural network and follow-up digital mammography", May 2018, p. 4. DOI: 10.1117/12.2304564.

[26] G. An, K. Omodaka, K. Hashimoto, S. Tsuda, Y. Shiga, N. Takada, T. Kikawa, H. Yokota, M. Akiba, and T. Nakazawa, "Glaucoma diagnosis with machine learning based on optical coherence tomography and color fundus images", *Journal of healthcare engineering*, vol. 2019, 2019.

[27] J. Berdahl, *Glaucoma: Symptoms, treatment and prevention*, Available at https://www.allaboutvision.com/conditions/glaucoma.htm, Jun. 2019.

[28] J. Brownlee, *What is deep learning?*, Available at https://machinelearningmastery.com/what-is-deep-learning/, 2019.

[29] A. Diaz-Pinto, S. Morales, V. Naranjo, T. Köhler, J. M. Mossi, and A. Navea, "Cnns for automatic glaucoma assessment using fundus images: An extensive validation", *Biomedical engineering online*, vol. 18, no. 1, p. 29, 2019.

[30] N. Donges, *What is transfer learning? exploring the popular deep learning approach*, Available at https://builtin.com/data-science/transfer-learning, 2019.

[31] P. Dwivedi, *Understanding and coding a resnet in keras*, Available at https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33, 2019.

[32] J. Jeong, *The most intuitive and easiest guide for convolutional neural network*, Available at https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480, 2019.

[33] L. Li, M. Xu, X. Wang, L. Jiang, and H. Liu, "Attention based glaucoma detection: A large-scale database and cnn model", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[34] *Types of glaucoma*, Available at https://www.glaucoma.org/glaucoma/types-of-glaucoma.php, Jun. 2020.

[35] A. Yadav, *Simple mathematics behind deep learning*, Available at https://towardsdatascience.com/simple-mathematics-behind-deep-learning-c38152c8b534, 2020.

[36] A. Amini, *Convolutional neural networks — mit 6.s191*, Available at https://www.youtube.com/watch?v=iaSUYvmCekI&t=285s.

[37] *Color fundus photography*, Available at https://ophthalmology.med.ubc.ca/patient-care/ophthalmic-photography/color-fundus-photography/.

[38] *Glaucoma*, Available at https://www.osmich.com/glaucoma-dearborn/.

[39] *Glaucoma*, Available at https://www.webmd.com/eye-health/glaucoma-eyes#1.

[40] *Glaucoma*, Available at https://www.aoa.org/patients-and-public/eye-and-vision-problems/glossary-of-eye-and-vision-conditions/glaucoma.

[41] M. Isaksson, *Four common types of neural network layers*, Available at https://towardsdatascience.com/four-common-types-of-neural-network-layers-c0d3bb2a966c.

[42] *Machine learning handbook*, Available at https://www.perceptilabs.com/resources/handbook.

[43] *Ndc-ivm: An automatic segmentation of optic disc and cup region from medical images for glaucoma detection*, Scientific Figure on ResearchGate. Available at https://www.researchgate.net/figure/Normal-and-glaucoma-aRected-eyes_fig1_312958793 [accessed 19 Jun, 2020].

[44] Stemplicity, *What is imagenet?*, Available at https://www.youtube.com/watch?v=gogV2wKKF_8&t=243s.

[45] *The jupyter notebook*, Available at https://jupyter.org/.

[46] *Transfer learning with a pretrained convnet*, Available at https://www.tensorflow.org/tutorials/images/transfer_learning.

[47] S.-H. Tsang, *Review: Inception-v3 — 1st runner up (image classification) in ilsvrc 2015*, Available at https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c.