PREDICTING DIABETIC RETINOPATHY SEVERITY VIA GENERATIVE ADVERSARIAL NETWORKS, CONVENTIONAL NEURAL NETWORK AND SUPPORT VECTOR MACHINES

STUDENT NAME: VIKRAM R SREEDHAR
SUPERVISOR NAME: DR. SUBHRAJYOTI MANDAL

A THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS OF LIVERPOOL JOHN MOORES UNIVERSITY

FEBRUARY 2020

2.5 cm

Intentionally left blank

3 cm

2 cm

TABLE OF CONTENTS

ACKNOWLEDGMENTS

My humble salutations to Gurumaharaj Sri Ramakrishna Paramahamsa for his divine grace and giving me that impulse for desiring to struggle and learning new aspects of knowledge. During his lifetime he had proclaimed “As long as I live, so long will I learn”. His life and teachings have guided me throughout and will always bear light and guidance in future also.

I would thank all my family members, friends, colleagues for bearing with me and my varied moods and providing me providence and support during the preparation of this report.

I would deeply acknowledge and thank my thesis supervisor Dr. Subhrajyoti Mandal in assisting me throughout this research work and guiding me rightfully in areas of Artificial Intelligence, Deep learning & Machine learning knowledge areas and computer programming.

ABSTRACT

Enough research on using new deep learning models like CNN, fractal analytics and Support vector machines have been used in prediction of diabetic retinopathy, glaucoma and other ocular diseases. Generative Adversarial Nets was one of the backward propagation algorithms which was developed by Ian Goodfellow and is being seriously considered by researchers as one of the best models for prediction and classification. In this paper have tried to use, conventional neural network, support vector machines and Generative Adversarial Nets (GAN) algorithms and their performance in trying to classify the ocular images for prediction of proliferative or non-proliferative to diabetic retinopathy. CNN performs better than SVM in terms of accuracy on train and test data. However, on Independent data set SVM returns a better accuracy than CNN. We obtained an extremely stable Generative Adversarial Network with adversarial and discriminator loss between the prescribed range.

LIST OF ABBEREVATIONS

CNN	Conventional Neural Networks
GAN	Generative Adversarial Networks or Nets
FA	Fractal Analytics
OCT	Optical Coherence Tomography
DR	Diabetic Retinopathy
SVM	Support vector machines
RNFL	Retinal nerve fibre layer
AUROC	Area under receiver operating characteristic curve
WFA	Wavelet Fourier Analysis
FFA	Fast Fourier Analysis
GPU	Graphical Processing Unit

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND FOR THE STUDY

There have been many studies on Glaucoma prediction using CNN, deep neural network, fractal analysis and support vector machine. Need to research whether new algorithms like GAN which will aid in faster accurate diagnosis of Glaucoma in comparison to other tested models of CNN and SVM. Also, to check by analyzing the Noise, Bias and Variance in above three models and using further method of prediction based on above model predictions to improve the overall performance and accuracy of prediction

1.2 PROBLEM STATEMENT

Investigate a diagnosis of diabetic Retinopathy by using imaging ophthalmic data with back propagation algorithms like SVM, CNN and GAN.

1.3 AIM AND OBJECTIVES

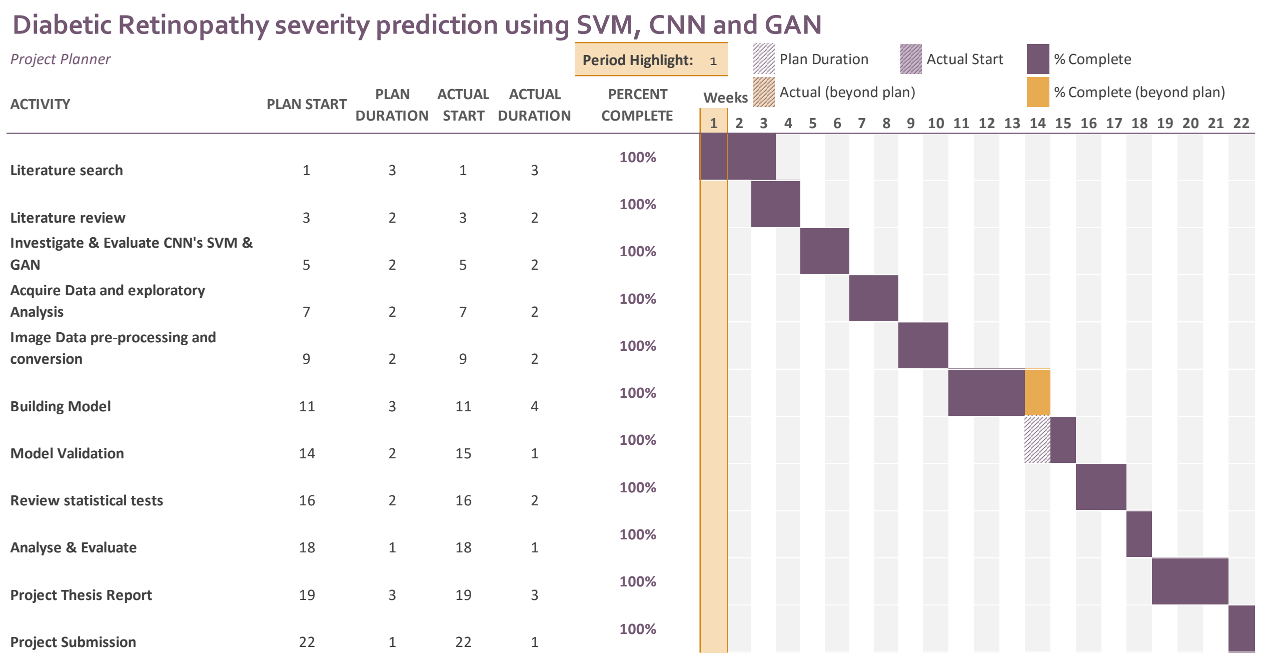
To investigate which algorithm is better in diagnosis of diabetic retinopathy prediction by using evaluation metrics like accuracy, recall, specificity, sensitivity and RUC for comparison.

1.4 SCOPE OF THE STUDY

Research will evaluate between which would be best back propagation model amongst conventional neural network, support vector machines and Generative Adversarial Nets in prediction of severity of diabetic retinopathy. Future research opportunity would be to build an ensemble model using all the three models or other back propagation models.

1.5 SUMMARY

Following is the progression in the entire research throughout the project.



Diabetic Retinopathy severity prediction using SVM, CNN and GAN

Project Planner

Period Highlight: 1

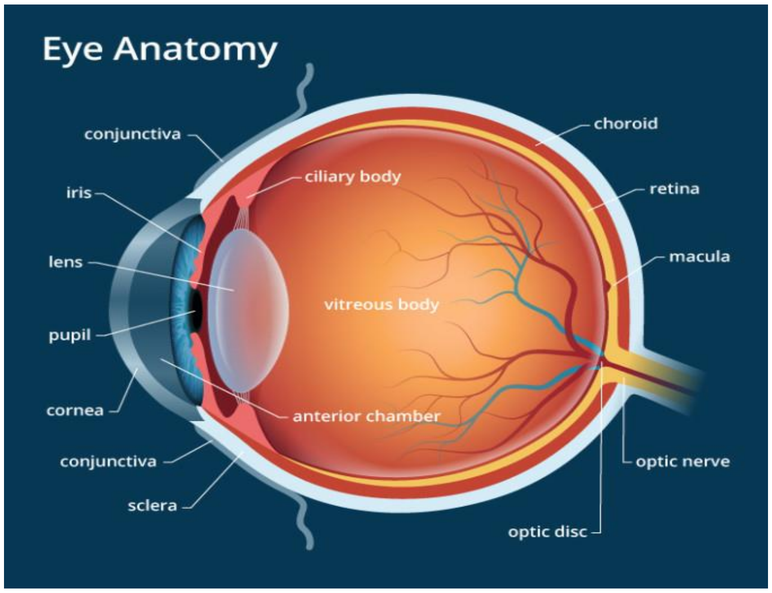
ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Literature search	1	3	1	3	100%
Literature review	3	2	3	2	100%
Investigate & Evaluate CNN's SVM & GAN	5	2	5	2	100%
Acquire Data and exploratory Analysis	7	2	7	2	100%
Image Data pre-processing and conversion	9	2	9	2	100%
Building Model	11	3	11	4	100%
Model Validation	14	2	15	1	100%
Review statistical tests	16	2	16	2	100%
Analyse & Evaluate	18	1	18	1	100%
Project Thesis Report	19	3	19	3	100%
Project Submission	22	1	22	1	100%

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

There have been many studies on prediction of ocular diseases using back propagation algorithms which is a approach of training the coefficients (weights) in multi-layered feed forward neural networks. Back propagation models have been used prominently by researches as it is fast, simple, easy to code and does not require too many parameters, apart from the input data. Disadvantages of these algorithms has been that it is quite sensitive to input data and noise in input data distorts the prediction ability. We will now review various research papers which have used these algorithms in prediction of ocular diseases.

2.2 BASICS OF EYE ANATOMY AND DIABETIC RETINOPATHY



(Image Courtesy – www.allaboutvision.com/resources/anatomy.htm)

Human eye works on the principles of light. Outward light is focused primarily by cornea. Iris controls the amount of light reaching the back of the eye by adjusting the pupil size. Eyes lens are behind the pupil and helps in automatic focus of vision for near and far of sight objects by adjusting like a camera focus. Light focused by cornea and crystalline lens and limited by iris and pupil reaches the retina, the light sensitive inner lining at back of the eye. Retina acts like image sensor, converts light to signals and transmits the signals to the visual cortex, part of the brain which controls our sight function. [1]

Diabetic retinopathy (DR) is most common diabetic eye disease leading of fatal blindness. It is damage in blood vessels in the retina as a result of Diabetes mellitus. Damage in blood vessels in retinal tissues results in leaking fluid and distorting vision. Milder form of diabetic retinopathy is known as non-proliferative and usually symptomless and proliferative retinopathy is advance stage resulting in abnormal blood vessels in retina. It is estimated that worldwide one third of patients with diabetes mellitus shows sign of DR. Possible complications associated with DR includes,
Vitreous hemorrhage – a newly formed blood vessels leaks into vitreous gel that fills the eye, stopping the light entering from retina.

Detached retina – scar tissue can pull away the retina from the back of the eye

Glaucoma – The normal flow of fluid in the eye may become blocked as new blood vessels form. Blockage cause build up in ocular pressure, increasing the risk of optic nerve damage and vison loss [2]

2.3 AUTOMATIC RECOVERY OF OPTIC NERVEHEAD GEOMETRY IN OPTICAL COHERENCE TOMOGRAPHY [3]

Optical coherence tomography (OCT) uses retroreflected light to provide micrometer – resolution, cross sectional scans of biological tissues. OCT has been useful in diagnosing, monitoring and studying glaucoma. Diagnosing glaucoma is difficult and once it progresses, neural tissues die, the nerve fiber layer thins and cup to disk ratio in eye increases. This paper presents the results from optic Nervehead segmentation and geometric characterization from OCT data. This paper presents development of autonomous algorithms based on a parabolic model of cup geometry and an extension of the Markov model to segment the retinal Nervehead surface, identify the choroid Nervehead boundary and identify the extent of optic cup. The results from this algorithm is compared with those of experienced ophthalmologist, reporting a correlation coefficient for cup diameter and disk diameter. [3]

The paper focuses on Retinal Vitreal boundary detection through a mathematical formulation and boundary detection algorithm. Boundary detection algorithm comprises four basic operations which is noise reduction, column wise edge detection, optimal threshold selection and final boundary extraction. The results were compared with the gold standard of benchmark which is experienced ophthalmologist's rating. Considerable mean, median and standard deviation of variances in pixel boundary were noted. [3]

This approach was first of sorts and instead of ophthalmologist's observation a calculation algorithm was built. Major limitations in this model were simplistic noise reduction algorithm than compared to current image pre-processing algorithms used in machine learning and significant reliance on radiologists OCT image generation. This method was bettered by future studies on machine and deep learning.

2.4 PREDICTION BASED ON FRACTAL FEATURE ON GLAUCOMA DETECTION AND PROGRESSION [4]

This paper investigates the use of fractal analytics (FA) as basis of multiclass prediction of progression of glaucoma. FA is applied to pseudo 2-D images converted from 1-D retinal nerve fibre layer (RNFL) data obtained from the eyes of normal subjects, and from subjects with progressive and non-progressive glaucoma. FA features are obtained using a box-counting method and a multifractional Brownian motion method that incorporates texture and multiresolution analyses. Sensitivity, specificity and AUROC are computed for FA features and for metrics obtained using wavelet-fourier analysis (WFA) and Fast Fourier Analysis (FFA). Features are then fed into SVM and Neural network (NN) models and correct rates of classification are arrived at. Classification rates are better in FA than compared to WFA and FFA. However, on computational time FFA takes more time than WFA and FFA as features extracted are minimal in latter two methods.   [4]

Data set of 96 subjects is been used in this analysis. Methodology of research is as follows:

a) Read 1 – D RFNL data
b) Segregate training and testing groups
c) Obtain fractal dimension using box counting and multifractional Brownian motion method
d) Append the fractal dimensions arrived with above mentioned both methods for feature fusion
e) Normalise the features
f) Apply Principal component analysis by computing covariance matrix and eigenvectors
g) Apply multiclass SVM and Neural network classifier
h) Get classification performance. [4]

With more advent in unstructured classification data higher algorithms required to solve
More unstructured data hence an opportunity to use CNN and GAN. Also, limitation of FFA and WFA as a method in feature conversion as they do not capture inherent randomness and irregularity of RFNL damage make further research possible in this area to use new back propagation methods in classification problems.

2.5 OPTICAL DISC LOCALIZATION USING SUPPORT VECTOR MACHINES [5]

Optic disc is one of the fundamental regions located in the internal retina that helps ophthalmologists in analysis and early diagnosis of many retinal diseases such as optic atrophy, neuritis, papilledema, ischemic optic neuropathy, glaucoma and diabetic retinopathy. This research provides an algorithm that provides a novel optic disc localization and segmentation technique that detects multiple candidate optic disc regions from fundus image using enhancement and segmentation. The algorithm provides a system that extracts hybrid feature set for each candidate region consisting of vessel based and intensity-based features which are finally fed to SVM classifier. Final decision of optic disc region is done after computing Manhattan distance from the mean of training data feature matrix. [5]

Methodology of research in this paper is as follows:

i) Image Pre processing
j) Blob detection
k) If blob available in multiple regions, then crop equidistant regions and extract features of vessel density, vessels orientation and intensity-based statistics and load it to SVM classifier
l) If multiple regions classified, then select the region with minimum Manhattan distance training feature matrix.   [5]

A Total of 1972 images from varied data sets was analysed and results were in the range of 96.75% to 100%.

A good accurate approach but algorithm proposed only focuses on OD localization and segmentation and thus leaves the diagnosis back to ophthalmologists. There has been advent of machine learning models which classifies the disease prediction and hence future research areas would be use newer algorithms in ocular disease prediction.

2.6 GLAUCOMA PREDICTION USING (CNN) [6]

The objective of this research paper is to use CNN to achieve the automatic detection of the disease. Experiments performed obtained an average accuracy of 99%.[6]

Total of 36 images were taken where in 22 were having glaucoma and balance 14 were that of a healthy eye. The paper also discusses about Glaucoma as a public health problem affecting approximately 6.7 million all over the world. Advent of technology has made the ophthalmic images quality improve over time, but the machines lack automatic detection of the disease. [6]

Sample size was limited in this approach and hence an opportunity to use higher sample size and other models of back propagation.

2.7 PATHOLOGICAL EVIDENCE EXPLORATION IN DEEP RETINAL IMAGE DIAGNOSIS [8]

In this paper GAN based method to synthesize pathological retinal image given the descriptor and binary vessel segmentation is proposed. [8]. To exploit the network interpretability in medical imaging, it is proposed to encode descriptor from the activated neurons that directly related to the prediction. Methodology of Koch's postulates that aim to identify the unknown pathogen was used. In addition, GAN based visualization method to visualize the pathological descriptor retinal image from an unseen binary vessel segmentation was used. Medical images generated from the model showed medical plausible symptoms as the reference image. Since pathological descriptor associated with individual lesion and spatial independent one could manipulate the position and identify the symptom. [8]

2.8 BACK PROPOGATION ALGORITHM

The Backpropagation algorithm is an approach for multilayer feed-forward networks in supervised learning in the field of deep learning neural networks

Feed-forward neural networks are inspired by the information processing of one or more neural cells, called a neuron. Dendrites sends the input signals to Neurons, which passes an electrical signal down to the body of the cell. The axon carries the signal out to synapses, which are the connections of a cell's axon to other cell's dendrites.[21]

In backpropagation fundamental proposition is to model a given function by modifying internal coefficients from input layers to produce an expected output coefficient at output layer. It is the whole quintessence of neural network being trained. The system is trained by using a supervised learning method, wherein the internal state is modified by knowing the incorrect result between the system's output and a known expected output from previous iterations or commonly known as epochs

As such, it requires a network architecture to be defined of one or more layers where one layer is fully linked together to the next layer. A standard network structure includes input layer, hidden layer, and an output layer. Inputs are modelled using randomly selected weights or coefficients and travel through the path of neural network by calculating the output for every neuron from input layer to hidden layer to output layer. Error which the difference between actual output and desired output is calculated in the process. As a process weights are readjusted and process of back tracking from output layer to hidden layer to input layer is done to minimize the error. This process is repeated until the desired output is realised. Backpropagation is widely used in classification and regression problems.

2.9 GENERATIVE ADVERSARIAL NETWORK (GAN) [7]

GAN's are deep neural architectures comprised of two networks competing against each other for domination and thus the name "Adversarial" [7]

To understand GAN, one must understand generative and discriminative algorithms. A discriminative algorithm could predict whether the image is real or fake or a message is a spam or not a spam. Spam is one of the labels and collection of words gathered from an e-mail and are the features that constitute the input data. When the problem is expressed mathematically, the label is called y and the features are called x. The formulation p(y|x) is used to mean the probability of y given x, which in this case mean the probability the email spam given the word it contains, or image is fake given the features and dimensions it contains. [7]

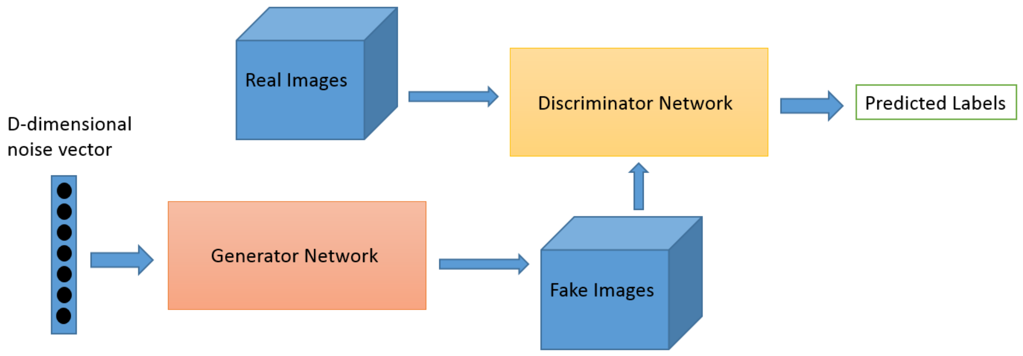
Generative algorithms work the opposite way. Instead of predicting a label given the features, it predicts the features given a certain label. The formulation in terms of probability would be p(x|y)

Here is how GAN works

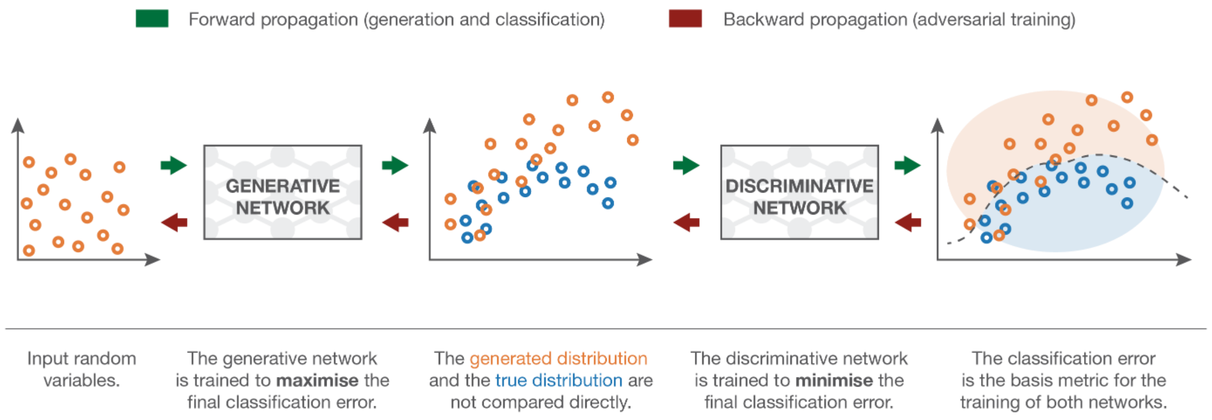
- The generator takes in random noise and returns an image.
- This generated image is fed into the discriminator alongside a stream of real images taken from the authentic observed dataset.

- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authentic image and 0 representing fake or counterfeit image.
- The discriminator is in a feedback loop with the authentic images' dataset, which we know.
- The generator is in a feedback loop with the discriminator. [7]

Pictorially GAN network can be represented as below.



(Image Courtesy – O'Reilly, [7])


Forward propagation (generation and classification)
Backward propagation (adversarial training)
GENERATIVE NETWORK
DISCRIMINATIVE NETWORK
Input random variables.
The generative network is trained to maximise the final classification error.
The generated distribution and the true distribution are not compared directly.
The discriminative network is trained to minimise the final classification error.
The classification error is the basis metric for the training of both networks.

(Image Courtesy – [8] – Joseph Rocca)Generative Adversarial Networks representation.

The generator takes simple random variables and noise as inputs and generate its own imaginary data or images. The discriminator with a classifier algorithm built within takes “true” and “generated” data or image and try to separate or distinguish them. The goal of the generator is to trick or deceive the discriminator (increase the classification delusion by mixing potentially to its maximum imagined data with observed ground data) and the goal of the discriminator is to distinguish between real and fake image of data.[8]

2.10 CONVULATIONAL NEURAL NETWORKS

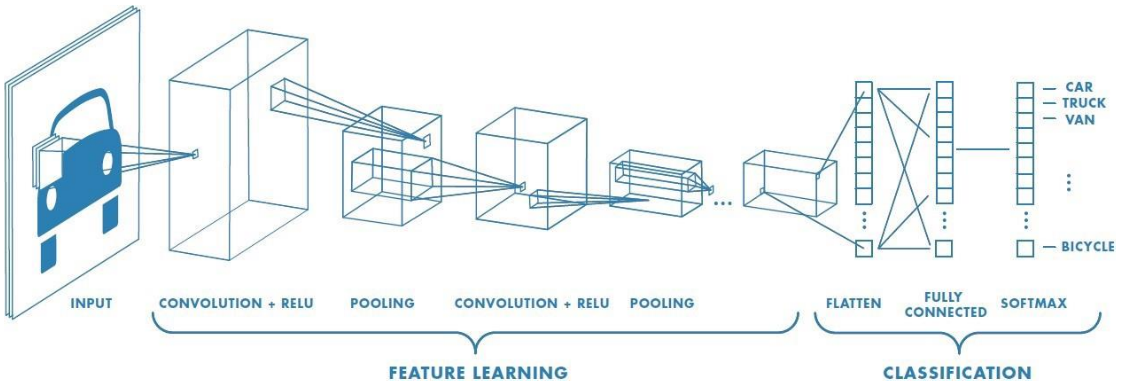


Image Courtesy [10] Artistic impression of CNN

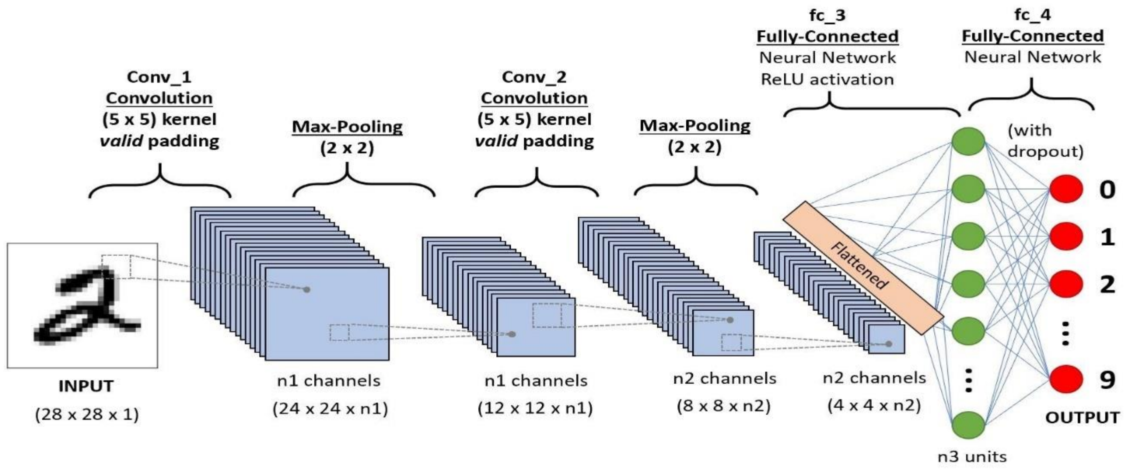


Image Courtesy [10]   A CNN framework to identify handwritten digits

Advancements in computer vision with deep learning has enabled machines to view as humans perceive knowledge for image recognition, classification and recommendation systems. [10]

A Convolutional Neural Network (CNN) is a deep learning algorithm which can process an input image, assign relevance (learnable weights and biases) to various features and dimensions in the image and be able to differentiate one from the other.

The framework of CNN is comparable to that of the associative paradigm of Neuron in the human brain

An Image is nothing but an array or matrix of picture elements or widely known as pixel values. Why then just not flatten the image and feed it to a Multi-Level supervised learning algorithm know as perceptron for classification purposes? Though not this approach is entirely correct, in basic binary images this method may show an average accuracy, while performing categorical classification with complex images having pixel dependencies this method will return nil or minimal accuracy in prediction. [10]

CNN can successfully capture the three dimensional and transient aspects in an image through invoking relevant filters. The role of CNN is to reduce the images into a form which is less complex to analyse without losing independent and reliant features which are critical for prediction of an image. The kernel moves many times because of stride length every time computing matrix multiplication operation over the image over which kernel is poised.

The filter moves to the right with a particular stride value till it examines the complete width until the entire image is transited. The kernel has the same depth as that of input image, in case of images which are other than grey scale, i.e. colour images. Matrix multiplication is executed, and all results are aggregated with the bias to give a flattened single depth channel convoluted or knotty feature output.

With diverse layers or strata of architecture in the CNN model low level as well as high level traits such as edges, colour orientation, gradients are captured resulting in understanding the images under consideration as a human brain would carry out.

Valid padding in the CNN framework is used to diminish dimensionality and Same padding is used to either enhance or sustain the dimensionality as per input image respectively.

Diminishing the structural size of the convolved elements is a major responsibility of the Pooling Layer. This is done to minimize the calculation intensity to process the image. Further through this process ascendant features which are rotational and positional unvaried are obtained for training the model effectively.

Two categories of pooling known as Max pooling and Average pooling are available. Max pooling returns the maximum value from the part of the image surfaced by the kernel and discards the noisy activations by performing de-noising (removing smudge and stains on an image) along with reducing the dimensionality. Average pooling returns the average value from the part of image surfaced by the kernel and only drops dimensionality without de-noising. [10]

Without an activation function neural network will be just same as linear regression models. With activation function neural networks would be able to learn and model with intricate data such as speech, image, audio, videos etc. Popular activation functions are Sigmoid or logistic, Hyperbolic tangent (Tanh) and Rectified Linear units (ReLu). ReLu is popular and almost all deep learning models use ReLu. It is popular as it avoids vanishing slope problem. Variation of ReLu function is leaky ReLu. The disadvantage of inputs become negative or tend to zero the slope function becomes zero in ReLu and backpropagation in network cannot be computed and learning stops. This is overcome by Leaky ReLu in terms of having a small positive gradient in negative area and thus enabling backpropagation computation and allowing the learning in network to continue. However, the disadvantage of Leaky ReLu is its predictions over negative inputs is not consistent.

The convolutional and the pooling layer all together form the n-th layer of a convolutional Neural Network. Number of such layers can be enhanced for apprehending detailed features of an image by trading off computational processing ability.

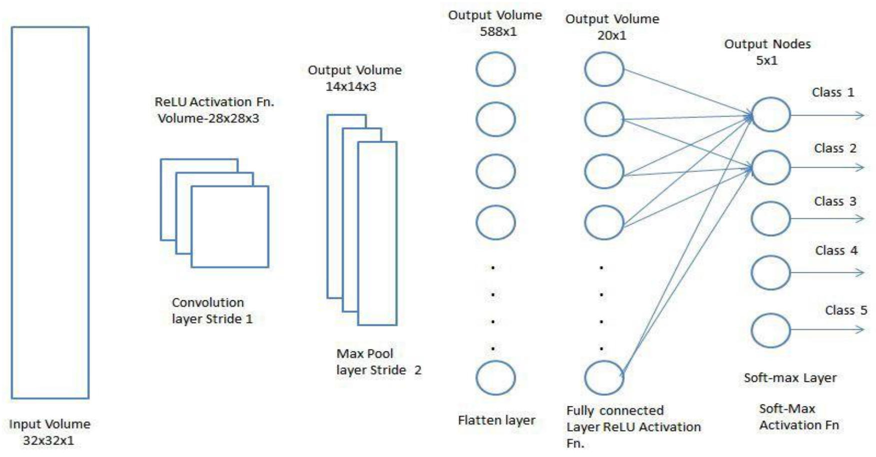


Image Courtesy [10] - Fully Connected Convolutional Neural Network

The fully connected layers at connection of all the activations in all the previous layer is squashed or flattened into a column vector at output layer. At every iteration or epochs of training the flattened output is fed to a feed-forward neural network and backpropagation is applied. Over the series of epochs, the model can distinguish between ascendant and few descendant features of an image.

2.11 SUPPORT VECTOR MACHINES

Support Vector Machine (SVM) is an advanced machine learning technique which has a unique way of solving complex problems such as image recognition, face detection, voice detection etc. SVMs solves the problem of nonlinearity through kernels.

By discovering the hyperplane classification is done that distinguishes the binary or more than two classes properly. They are emanated on the conception of decision planes that define decisive boundaries. An illustration is shown below.

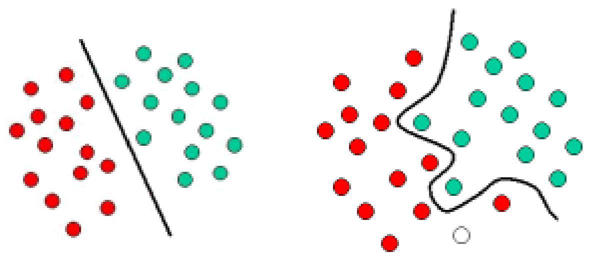


Image Courtesy [12]

Above we see an example of linear classifier and other one which is more complex than a line and requires a curvature line to set apart green and red objects.

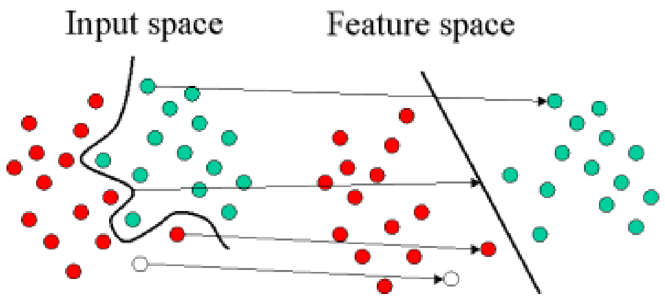


Image Courtesy [12]

We see from the above figure that original objects which is reorganized and segregated using a set of mathematical functions know as kernels. The process of rearranging or reorganizing the objects is known as transformation. SVM is a classifier model that executes classification jobs by forming hyperplanes in a multidimensional space that insulates different class labels.

In actuality the way SVM works is defined by the Maximal margin classifier which is a hypothetical classifier. The numeric input independent variables (x) in the data (the columns) form a multi-dimensional space. To illustrate, if we have two input independent variables than we have a two-dimensional space. For example

$$B0 + B1 * X1 + B2 * X2 = 0$$

Where B0 is an intercept found by learning algorithm and the coefficients B1 and B2 are slope of the line, and two input independent variables are X1 and X2. The distance between the closest data points and the separating line is referred as the margin. The line that as the largest margin is the best or optimal line than can separate the two classes. This is known as the Maximal-

Margin hyperplane. It is computed as the perpendicular distance from the line to only the closest points. These points are known as the support vectors. They define the hyperplane. [13]

In practice the actual data is intricate and tricky and cannot be isolated with precision with a hyperplane. The limitation of maximizing the margin of the line that segregates or isolates the classes should not be stringent. This is known as the soft margin classifier. This change allows some of the specks in the training data to contravene the dividing or segregating line. A tuning parameter is introduced known as C that defines the weight of the movement allowed across all the dimensions. The larger the value of C the more infractions to the hyperplane is enabled.

SVM algorithm is carried out by using Kernels. A Liner kernel will transform the problem by using a liner algebra algorithm and learning of the hyperplane for classification. To separate classes which are complex and curved, complex kernels such as polynomial or radial kernel are used.

The polynomial kernel allows for curved lines in the input space. A typical polynomial kernel equation is specified as

$$K(x, xi) \ = \ 1 + sum(x * xi)\text{^}d$$

Where the degree of the polynomial is specified by d. If d=1 then kernel is liner kernel. [13]

A radial kernel can create more sophisticated fields with the feature space like closed polygons in two-dimensional space. A sample equation of radial kernel is specified as

$$K(x, xi) \ = \ exp(-gamma \ * \ sum(x - xi\text{^}2))$$

Gamma is the value between 0 and 1 and defines the parameter for learning algorithm. [13]

2.12 SUMMARY

We have examined and reviewed different research papers which have elaborated research methods done in prediction of ocular diseases using image geometry through CNN’s, SVM, GAN and fractal analytics and specific method’s efficacy in prediction of disease condition in retina.

We have also examined theory behind back propagation algorithms and how convolutional networks, Generative Adversarial Nets and Support Vector Machines work.

Further, in this paper have tried to research comparative approach by conventional neural network, support vector machines and Generative Adversarial Nets (GAN) algorithms and their performance in trying to classify the ocular images for prediction of proliferative or non-proliferative disease condition of diabetic retinopathy

CHAPTER 3: RESEARCH METHODOLOGY – DATA PREPERATION

3.1 DATA SET

Data set was available on Kaggle which has 65k images. Basic data definition is as follows

The images in the dataset are from varied models of camera and image processing hardware, which can alter the visual look of left vs. right eye. Few images are anatomical representation of retina (we see macula on the left side, to the right side of right eye we see the optic nerve). Others are exhibited as it would appear or seen through condensing lens of a microscope (i.e. image is inverted, as it would appear in a normal examination of an eye). We can know if an image is inverted by two means: [22]

If the central area which is usually small and dark known as macula is slightly higher than the midline through the optic nerve, than the image is inverted. If the midline of the optic nerve is higher than the macula, then the image is not inverted. [22]

If there is a scratch on the side of the image (square, triangle, or circle) then the image is not inverted. The image is inverted if there is no scratch. [22]

Like in any actual situations data set has noise such as smudges or stains in both the images and labels. Images may have objects, which are out of focus, which are underexposed, or which are overexposed. In spite of noise and variation being present, we have to ensure we develop a robust algorithm that can be stable and generalized. [22]

On a level of 0 to 4, an ophthalmologist has graded the presence and severity of diabetic retinopathy in each image according to the following scale:

0 - No Diabetic Retinopathy

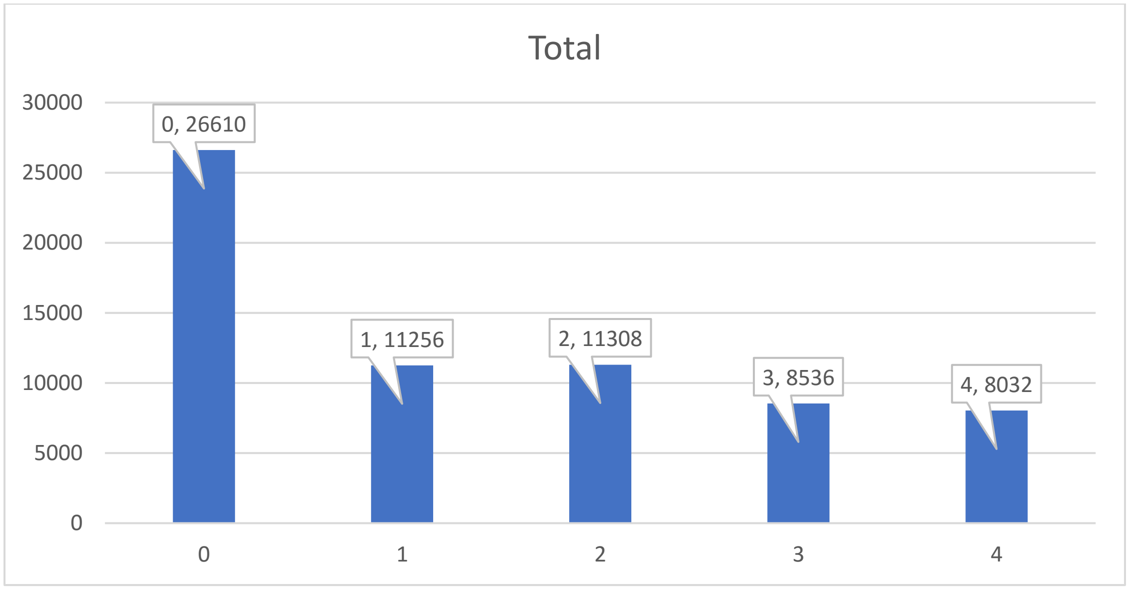
1 – Mild Diabetic Retinopathy

2 – Moderate Diabetic Retinopathy

3 – Severe Diabetic Retinopathy

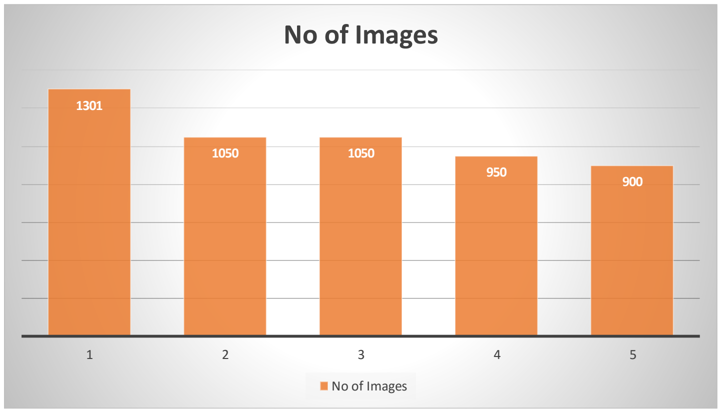
4 - Proliferative Diabetic Retinopathy [22]

3.2 EXPLORATORY DATA ANALYSIS OF TOTAL IMAGES IN KAGGLE DATASET AND IMAGES USED FOR THE PURPOSE OF ACHIEIVING RESEARCH OBJECTIVE

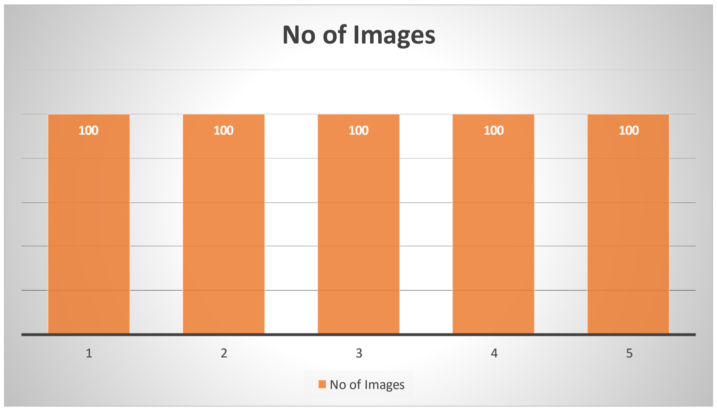


- A Total of 65742 images with 26610 images of label 0, 11256 images of label 1, 11308 images of label 2, 8536 images of label 3 and 8032 images of label 4.

- For the purpose of this thesis a total of 5251 images randomly selected from above 65742 images was used as train data for the purpose of achieving research objective. 1301 images of label 0, 1050 images of label 1, 1050 images of label 2, 950 images of label 3 and 900 images of label 4 were used in CNN, SVM and GAN training of models.



- For the purpose of this thesis a total of 500 images randomly selected from above 5251 training images was used as test data for the purpose of achieving research objective. 100 images of label 0, 100 images of label 1, 100 images of label 2, 100 images of label 3 and 100 images of label 4 were used in CNN, SVM and GAN training of models.



3.3 DATA/IMAGE PRE-PROCESSING

Input Images given in the data set were 512*512*3 dimension. Processing this quality of data on a normal CPU was a challenge and hence the data was transformed to size of 28*28*3 dimension. EBImage package was used in R for image pre-processing.

EBImage executes common purpose ability for image processing and analysis. EBImage offers techniques to slice cells and obtain numerical cellular descriptors in situations of microscopy based cellular trails which are high throughput images. Using the R programming language and with the use of other techniques, commands and packages available in the R program, EBImage facilitates automation of aforementioned activities such as statistical modelling, signal processing visualization and, machine learning with image data.[14]

Basic *EBImage* enables reading, writing, and displaying of images in R environment. Images are read using the command readImage, which takes as input an Uniform Resource Locator (URL) or a name of the image file. File formats of jpeg, png and tiff are supported by *EBImage*. RBioFormats package provides support and complements the above list by offering assistance for a much broader range of file formats including custodian microscope image data and bigdata. The image of a proliferative retina with label 4 which we loaded can be visualized by the command called display. [14]

Instead, the image can be displayed using R’s built in function called plot by calling display with a useful command approach known as “raster". The picture is then shown on the current system. This enables to add text labels effortlessly and bring together image data with other plotting abilities. [14]

Some of the image displays are displayed below. Below image is of retina with label 0 which is no Diabetic Retinopathy

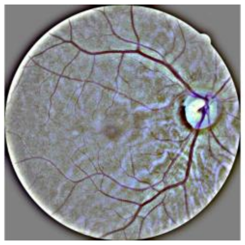
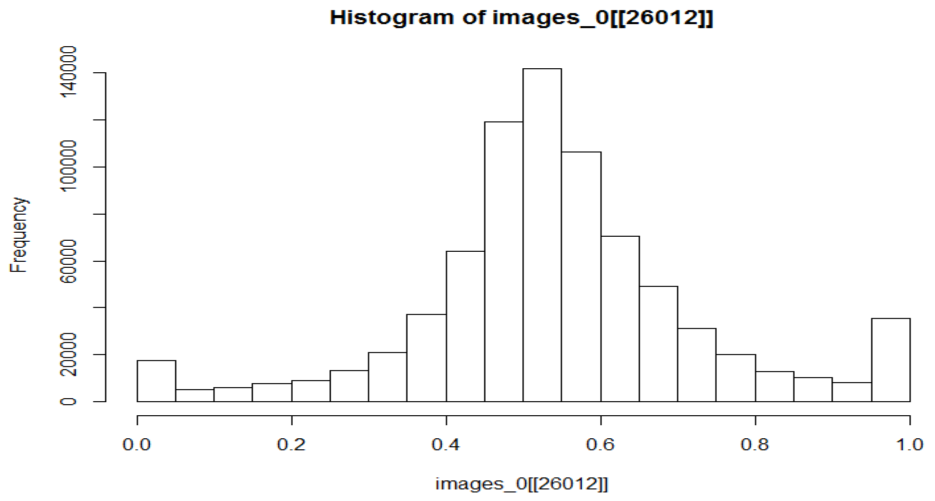


Image data representation is shown as below. *EBImage* uses a specific method known as class Image to save and handle image data. It facilitates in extending the R environment’s base class array, and all *EBImage* functions can also be directly called on arrays and matrices. Let us now see how the internal structure of an Image object of above retina image is stored by EBImage.[14]

```
 Image
 colorMode    : Color
 storage.mode : double
 dim          : 512 512 3
frames.total : 3
frames.render: 1
imageData(object)[1:5,1:6,1]



           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 0.5019608 0.5019608 0.4980392 0.4862745 0.4784314 0.4784314
[2,] 0.5843137 0.5686275 0.5372549 0.5058824 0.4862745 0.4862745
[3,] 0.5647059 0.5529412 0.5215686 0.4862745 0.4745098 0.4862745
[4,] 0.5450980 0.5490196 0.5333333 0.4980392 0.4705882 0.4784314
[5,] 0.5490196 0.5568627 0.5490196 0.5137255 0.4823529 0.4823529
```

The “frames.total” and “frames.render” parameters shown in above summary of image corresponds to the total number of frames contained in the particular object (since the image read is colour image it shows frame as 3 which means RGB and in case of grey image it would show frame as 1), and to the number of rendered frames. [14]

The distribution of pixel magnitudes of a particular image can be plotted on a histogram, and their spread examined invoking the range function. Below we see a histogram plotted with pixel values for label 0 No diabetic retinopathy image of retina.



A common pre-processing action involves cleansing the images by eliminating local residues or noise which are smudges and stains on images through a process known as smoothing. An automatic method is to define a frame of a selected size around each pixel and take the mean of the values within that precinct. After employing this approach or method to all pixels, a new, smoothed and denoised image is generated. Mathematically, this can be expressed as

$$f'(x, y) = \frac{1}{N} \sum_{s=-a}^{a}\sum_{t=-a}^{a} f(x+s, y+t)$$

where $f(x,y)f(x,y)$ is the numerical value of the pixel at a point $(x,y)(x,y)$, and aa defines the window or frame size, which is $2a+12a+1$ in each direction. $N=(2a+1)2N=(2a+1)2$ is the number of pixels averaged over, and $f'f'$ is the new, denoised or smooth image.[14]

More commonly, a weighted average can be swapped with the moving average, using a weight operate know as ww, which normally has the highest value at the window or pixel framework midpoint ($s=t=0s=t=0$) and then drops towards the edges.

$$(w * f)(x,y) = \sum_{s=-\infty}^{+\infty} \sum_{t=-\infty}^{+ infty} w(s,t)\, f(x+s,y+s)$$

For notational ease, it is shown to that he summations spread from -infinity to +infinity even if in practice the totals are finite and ww has only a limited number of non-zero values. Basically, we can assume of the weight function ww as another image, and this operation is known as the convolution or intricacy of the images ff and ww, indicated by the the symbol $**$. Convolution is a linear operation in the sense that $w*(c1f1+c2f2)=c1w*f1+c2w*f2w*(c1f1+c2f2)=c1w*f1+c2w*f2$ for any two images $f1f1$, $f2f2$ and numbers $c1c1$, $c2c2$. [14]

Another method to execute reduction of noise is to employ a median filter, which is a technique of non-linearity as against the low pass linear filter explained in the previous section. Median filtering is an appropriate and efficient technique in the case of images having smudge and stains as noise and has the benefit of removing smudges or stains while retaining edges. [14]

The local median filter is actioned by investigating the image by each pixel, substituting each pixel by the median on of its precinct inside a window or framework of specified size. This method of filtering is supported in *EBImage* by the command known as medianFilter. [14]

Noise reduction on the data used for modelling was done by using Median filter.

3.4 DATA/IMAGE TRANSFORMATION

Image content occurs at variety of scales. A larger scale means image of lower resolution and high-resolution image has finer scales. Neural networks capture more global features while using lower resolution images, but downside is it loses finer features, but advantages would be faster computing and training of data. Contrary high-resolution image captures finer features but loses out on global features. Therefore, in view of trade-off between the both the input image of 512*512*3 was reduce to 28*28*3. Feature reduction of images has been done to achieve following:

- Less useful information is discarded by the CNN
- Reasonable training time is achieved
- Constraints in terms of working on a CPU with 32 GB RAM and not able to work on GPU

3.5 SUMMARY

The images were resized and reshaped from 512*512*3-dimensional space to 28*28*3 of all labels of diabetic retinopathy of retina images. Each label of images was row binded individually and all the labels were bound and consolidated in an array format. Dependent categorical variables for each level was defined and also bounded into array format of data. Array reshape on dependent variables with pixel values was reshaped to (-1,28,28,3) for the data to be read to convolutional neural nets. Same processed and finalized data set was used for building Generative Adversarial Nets.

Data was transformed to data frame format with labels of each image pasted for the purpose of reading of data for Support Vector Machines modelling.

Train Label Y (dependent variable) and Test Label Y (dependent variable) were created which specified the category class of the image as 0,1,2,3,4 which meant No, mild, moderate, severe or proliferative condition of diabetic retinopathy.

One hot encoding on dependent variables were done for train and test label y to convert categorical data to integer data. This facilitates ML algorithms to execute predictions

CHAPTER 4: ANALYSIS, DESIGN AND MODEL ARCHITECTURE

4.1 CONVULATIONAL NEURAL NETWORKS FRAMEWORK AND DESIGN

Hyper parameter tuning of the model architecture to be deployed for CNN was done using keras, Boruta and psych packages in R. Tuning run on dense units' kernel size, dropout layers, epochs and batch size were set as parameter to return the ideal model architecture.

Following was the final CNN model architecture which as arrived at after hyperparameter tuning results. Relu activation was used in convolutional layers and in final dense layer soft maximisation was used.

```
# Model Architecture

model <- keras_model_sequential()

model %>%
  layer_conv_2d(filters = 32,
                kernel_size = c(3,3),
                activation = 'relu',
                input_shape = c(28,28,3)) %>%
  layer_conv_2d(filters = 32,
                kernel_size = c(3,3),
                activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_conv_2d(filters = 64,
                kernel_size = c(3,3),
                activation = 'relu') %>%
  layer_conv_2d(filters = 64,
                kernel_size = c(3,3),
                activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_flatten() %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dropout(rate=0.25) %>%
  layer_dense(units = 5, activation = 'softmax') %>%

  compile(loss = 'categorical_crossentropy',
          optimizer = 'adam',
          metrics = c('accuracy'))
summary(model)
```

Adaptive moment estimation or “adam” was used as the optimizer. It calculates flexible learning rates for each parameter. It stores both exponentially decaying average of past squared gradients, and exponentially decaying average of past gradients M(t), similar to momentum. [15] It is an update over Rmsprop and in a way combines stochastic gradient descent (SGD) and Rmsprop

M(t) and V(t) are values of the first moment which is the *Mean* and the second moment which is the *uncentered variance* of the *gradients* respectively. [15]

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}.$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

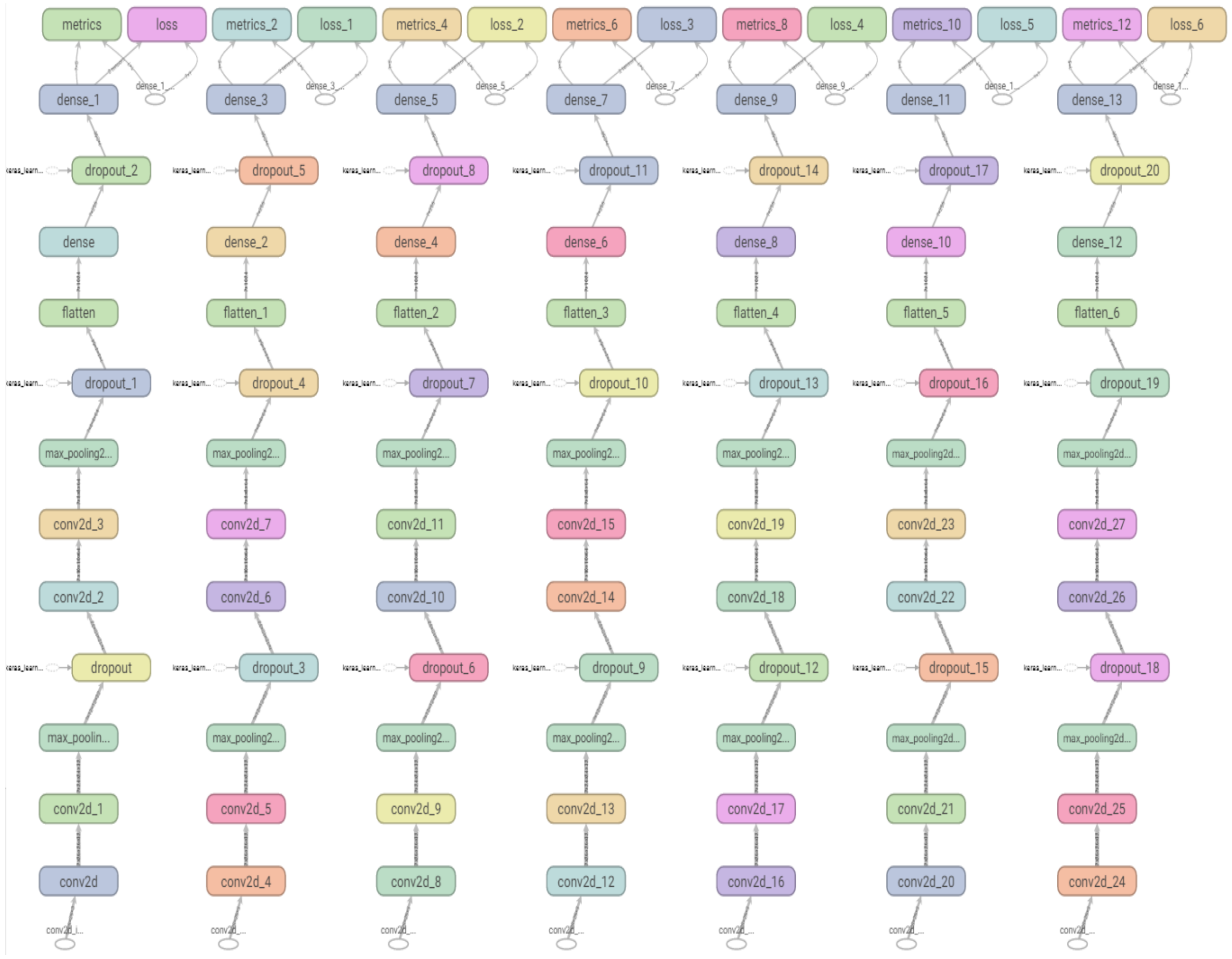
Then the final mathematical code for the Parameter update is

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$

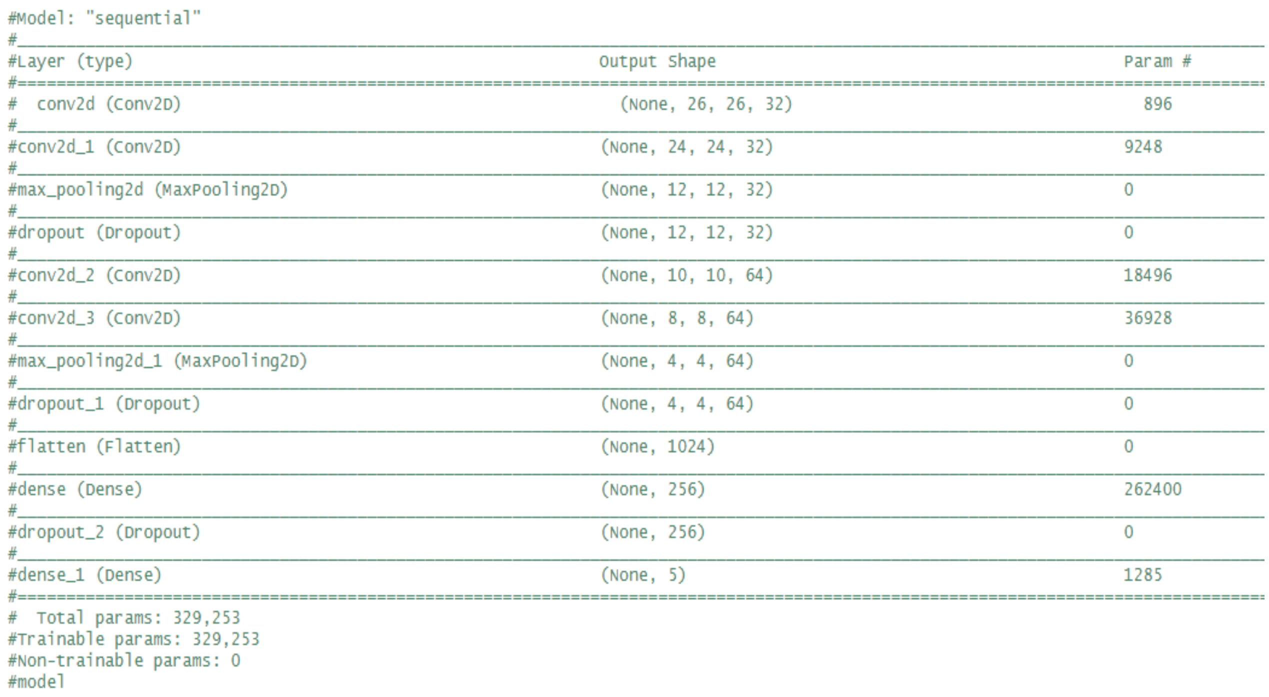
Adam works well in actual world and relates favourably to other adaptive learning-method algorithms as it unites very fast and the speed of learning is superiorly fast for the Model and is pretty quick and competent and also it addresses every problem that is encountered in other optimization techniques such as slow convergence , disappearing Learning rate or high variance in the parameter updates leading to inconsistent Loss function [15]

Irrespective of superior training time, Adam still does not do well and arrive best image classification solutions on certain familiar CIFAR datasets and underperforms with Adadelta and Rmsprop optimizers.

The schematic representation of CNN model architecture implemented is shown as per the below figure.



The summary of the model with output shapes and parameters are show below:



```
#Model: "sequential"
#____________________________________________________________________________________________________
#Layer (type)                                 Output Shape                              Param #
#====================================================================================================
#  conv2d (Conv2D)                              (None, 26, 26, 32)                        896
#____________________________________________________________________________________________________
#conv2d_1 (Conv2D)                            (None, 24, 24, 32)                        9248
#____________________________________________________________________________________________________
#max_pooling2d (MaxPooling2D)                 (None, 12, 12, 32)                        0
#____________________________________________________________________________________________________
#dropout (Dropout)                            (None, 12, 12, 32)                        0
#____________________________________________________________________________________________________
#conv2d_2 (Conv2D)                            (None, 10, 10, 64)                        18496
#____________________________________________________________________________________________________
#conv2d_3 (Conv2D)                            (None, 8, 8, 64)                          36928
#____________________________________________________________________________________________________
#max_pooling2d_1 (MaxPooling2D)               (None, 4, 4, 64)                          0
#____________________________________________________________________________________________________
#dropout_1 (Dropout)                          (None, 4, 4, 64)                          0
#____________________________________________________________________________________________________
#flatten (Flatten)                            (None, 1024)                              0
#____________________________________________________________________________________________________
#dense (Dense)                                (None, 256)                               262400
#____________________________________________________________________________________________________
#dropout_2 (Dropout)                          (None, 256)                               0
#____________________________________________________________________________________________________
#dense_1 (Dense)                              (None, 5)                                 1285
#====================================================================================================
#  Total params: 329,253
#Trainable params: 329,253
#Non-trainable params: 0
#model
```

Various layers have been added to develop a CNN Model. The input layer is the network has 28*28*3 dimensions based on height and width of the images. Since we are using colour images the colour channel is RGB.

The first input layer has an image with dimensions of 28*28*3. To obtain the output shape, we subtract three from kernel_size from height of input image which is 28 and add 1. The final output shape becomes 26 * 26 * 32 (where 32 is output filters). The output shape has reduced in height and width but has gained greater depth. Therefore, total parameters are 3*3*3*32+32 = 896, where 3*3 is the kernel_size, 3 is the number of channels for the image. 32 to output filters is added as 32 bias terms.

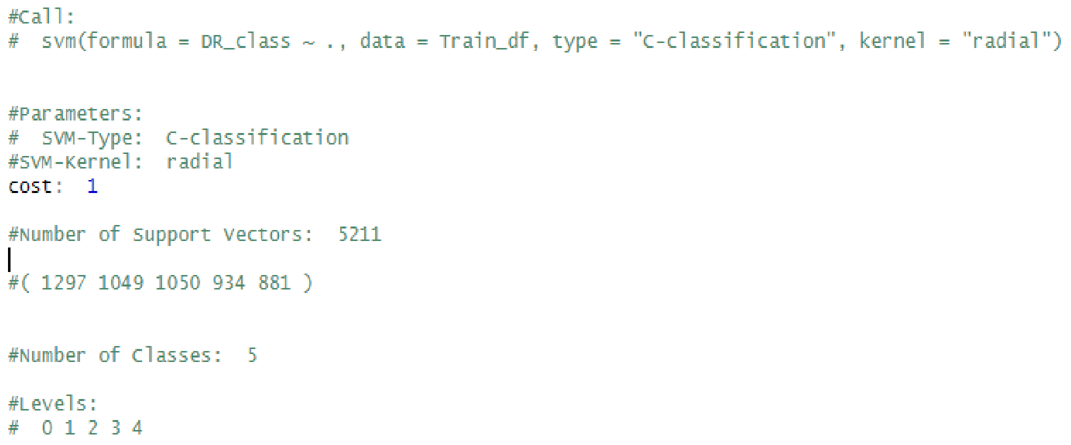
The output of second convolutional network is 9248 parameters. The next layer is the pooling layer, which is usually placed after the convolutional layer and performs a down-sampling operation. This helps in reducing computational time and to reduce overfitting. The pooling layer is followed by dropout layer to which no new parameters are added.

Then we have 3rd and 4th convolutional network with 18496 and 36928 parameters. The next layer is the pooling layer followed by the dropout layer with no parameters added. In the flattened layer, we go from 3 dimensions (4*4*64) to one dimension by multiplying the three to numbers to obtain 1024. This is followed by a densely connected layer with 256 units. The number of parameters here can be obtained as 1024*256+256 = 262400. This is followed by another dropout layer to avoid over fitting and no parameters are added here. Finally, we have the last layer, which is a densely connected layer with 5 units representing 5 classification labels. The number of parameters here is 256*5+5 = 1285 parameters. The total number of parameters is thus 329253 parameters. We are using relu activation for hidden layers and softmax for the output layer.

4.2 SUPPORT VECTOR MACHINES FRAMEWORK AND DESIGN

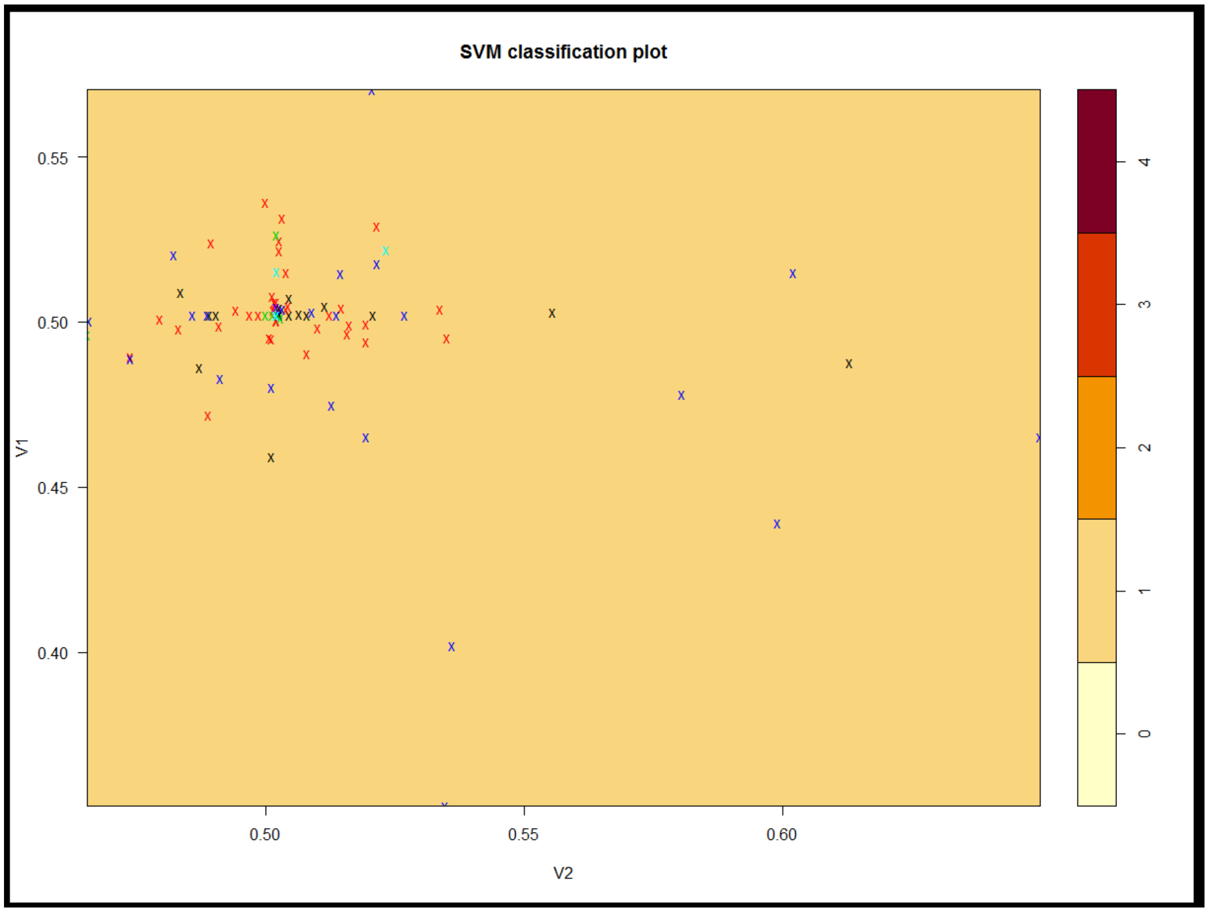
Hyper parameter tuning with 5-fold cross validation was done with sigma values from 0.1 to 0.5 with sequential tuning by 0.01 and c values from 1 to 5 with sequence of 1. Based on hyper parameter tuning results below SVM model was implemented.

An SVM model with radial kernel was implemented. Summary of the model is as follows



```
#Call:
#  svm(formula = DR_class ~ ., data = Train_df, type = "C-classification", kernel = "radial")


#Parameters:
#  SVM-Type:  C-classification
#SVM-Kernel:  radial
cost:  1

#Number of Support Vectors:  5211

#( 1297 1049 1050 934 881 )


#Number of Classes:  5

#Levels:
#  0 1 2 3 4
```

A total of 5211 support vectors in the model. A schematic architecture of SVM classifier model for 2 of variables of the total 2352 independent variables is shown in below illustration.

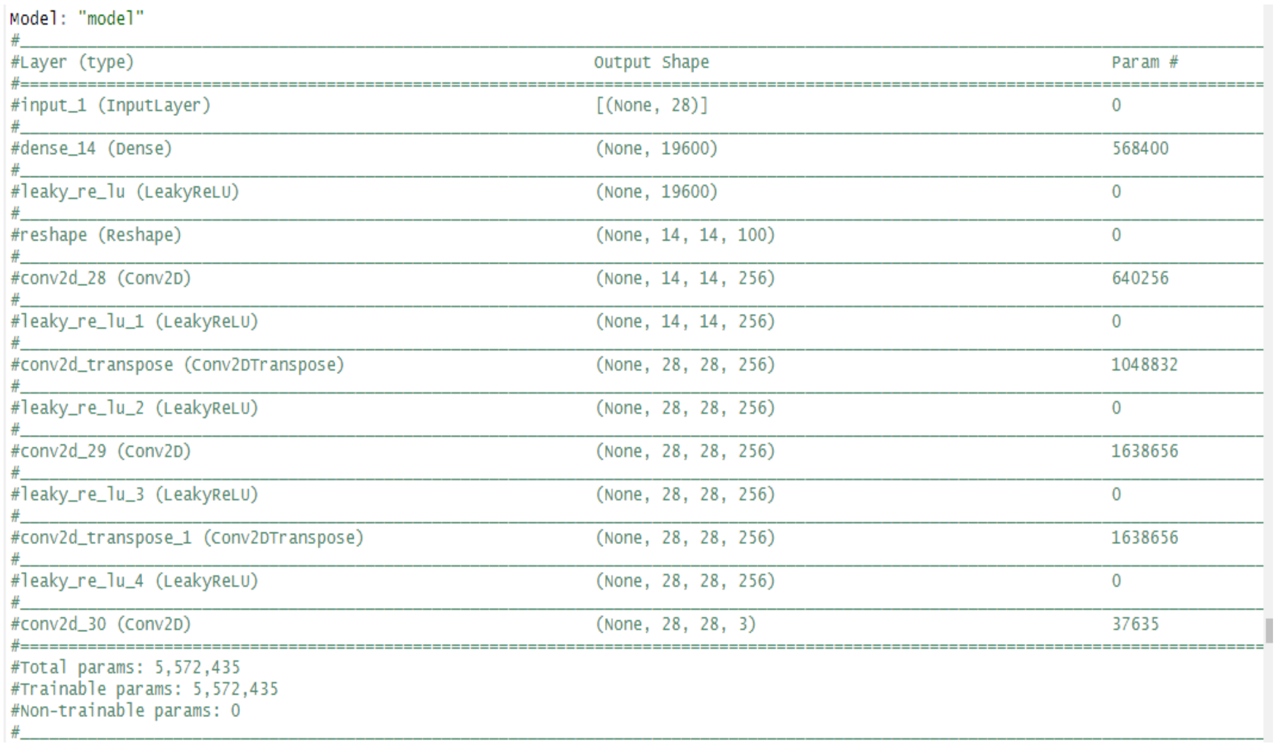


4.3 GENERATIVE ADVERSARIAL NETWORK FRAMEWORK AND DESIGN

Following is the architecture of Generator input and Output model Architecture. Tanh activation and same padding has been used. Length(l), height (h), width(w) and channels (c) has been defined as per the resized image resolution of 28*28*28*3 for defining Generator input model.

```
l <- 28
h <- 28
w <- 28
c <- 3

gi <- layer_input(shape = l)

go <- gi %>% layer_dense(units = 100 * 14 * 14) %>%
  layer_activation_leaky_relu() %>%
  layer_reshape(target_shape = c(14, 14, 100)) %>%
  layer_conv_2d(filters = 256,
                kernel_size = 5,
                padding = "same") %>%
  layer_activation_leaky_relu() %>%
  layer_conv_2d_transpose(filters = 256, kernel_size = 4,strides = 2, padding = "same") %>%
  layer_activation_leaky_relu() %>%
  layer_conv_2d(filters = 256,
                kernel_size = 5,
                padding = "same")%>%
  layer_activation_leaky_relu() %>%
  layer_conv_2d_transpose(filters = 256, kernel_size = 5,padding = "same") %>%
  layer_activation_leaky_relu() %>%
  layer_conv_2d(filters = c,
                kernel_size = 7,
                activation = "tanh",
                padding = "same")
g <- keras_model(gi, go)
```

Summary of generator network model with output shape and parameters are as follows:



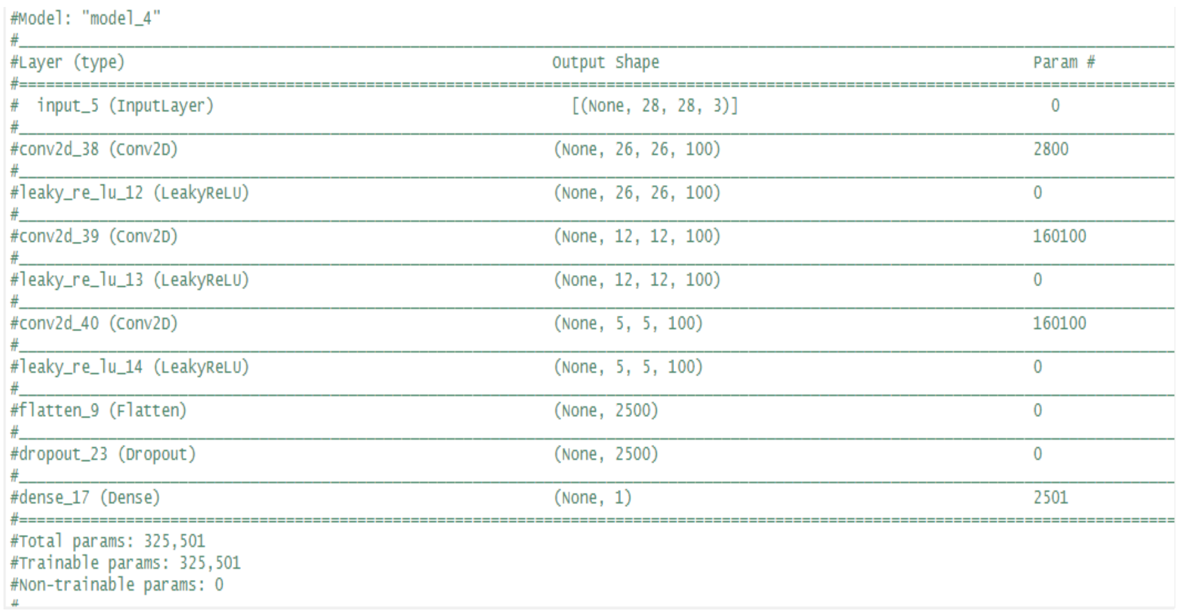
```
Model: "model"
#_________________________________________________________________________________________
#Layer (type)                          Output Shape                      Param #
#=========================================================================================
#input_1 (InputLayer)                  [(None, 28)]                      0
#_________________________________________________________________________________________
#dense_14 (Dense)                      (None, 19600)                     568400
#_________________________________________________________________________________________
#leaky_re_lu (LeakyReLU)               (None, 19600)                     0
#_________________________________________________________________________________________
#reshape (Reshape)                     (None, 14, 14, 100)               0
#_________________________________________________________________________________________
#conv2d_28 (Conv2D)                    (None, 14, 14, 256)               640256
#_________________________________________________________________________________________
#leaky_re_lu_1 (LeakyReLU)             (None, 14, 14, 256)               0
#_________________________________________________________________________________________
#conv2d_transpose (Conv2DTranspose)    (None, 28, 28, 256)               1048832
#_________________________________________________________________________________________
#leaky_re_lu_2 (LeakyReLU)             (None, 28, 28, 256)               0
#_________________________________________________________________________________________
#conv2d_29 (Conv2D)                    (None, 28, 28, 256)               1638656
#_________________________________________________________________________________________
#leaky_re_lu_3 (LeakyReLU)             (None, 28, 28, 256)               0
#_________________________________________________________________________________________
#conv2d_transpose_1 (Conv2DTranspose)  (None, 28, 28, 256)               1638656
#_________________________________________________________________________________________
#leaky_re_lu_4 (LeakyReLU)             (None, 28, 28, 256)               0
#_________________________________________________________________________________________
#conv2d_30 (Conv2D)                    (None, 28, 28, 3)                 37635
#=========================================================================================
#Total params: 5,572,435
#Trainable params: 5,572,435
#Non-trainable params: 0
#_________________________________________________________________________________________
```

The generator network shows the output's shape and the number of parameters for each layer. Note that the final output shape is 28*28*3. The fake images that will be generated will have these dimensions. Overall for this network, we have 5572435 parameters.

Following is the architecture of Discriminator input and Output model Architecture. Sigmoid activation has been used. Height (h), width(w) and channels (c) has been defined as per the resized image resolution of 28*28*3 for defining Discriminator input model. Discriminator network will be used for classifying fake and real images. In the last layer of the discriminator output (do) have specified the activation function and the units as 1, since an image is differentiated as either real or fake.

```
di <- layer_input(shape = c(h, w, c))
do <- di %>%
  layer_conv_2d(filters = 100, kernel_size = 3) %>%
  layer_activation_leaky_relu() %>%
  layer_conv_2d(filters = 100, kernel_size = 4,strides = 2) %>%
  layer_activation_leaky_relu() %>%
  layer_conv_2d(filters = 100, kernel_size = 4,strides = 2) %>%
  layer_activation_leaky_relu() %>%
  layer_flatten() %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 1, activation = "sigmoid")
d <- keras_model(di, do)


d %>% compile(loss = "binary_crossentropy",
        optimizer = optimizer_rmsprop(lr = 0.0008,clipvalue = 1.0,decay = 1e-8))
```

Summary of the discriminator model with parameters is as follows



```
#Model: "model_4"
#________________________________________________________________________________
#Layer (type)                          Output Shape                   Param #
#================================================================================
#  input_5 (InputLayer)                  [(None, 28, 28, 3)]            0
#________________________________________________________________________________
#conv2d_38 (Conv2D)                    (None, 26, 26, 100)            2800
#________________________________________________________________________________
#leaky_re_lu_12 (LeakyReLU)            (None, 26, 26, 100)            0
#________________________________________________________________________________
#conv2d_39 (Conv2D)                    (None, 12, 12, 100)            160100
#________________________________________________________________________________
#leaky_re_lu_13 (LeakyReLU)            (None, 12, 12, 100)            0
#________________________________________________________________________________
#conv2d_40 (Conv2D)                    (None, 5, 5, 100)              160100
#________________________________________________________________________________
#leaky_re_lu_14 (LeakyReLU)            (None, 5, 5, 100)              0
#________________________________________________________________________________
#flatten_9 (Flatten)                   (None, 2500)                   0
#________________________________________________________________________________
#dropout_23 (Dropout)                  (None, 2500)                   0
#________________________________________________________________________________
#dense_17 (Dense)                      (None, 1)                      2501
#================================================================================
#Total params: 325,501
#Trainable params: 325,501
#Non-trainable params: 0
#
```

Here the output of the first layer is 28*28*3 in size, which matches the dimensions of the fake and real images. Total parameters is 325,501.

For model compiling optimizer being used is RMSprop. A detailed description of RMSprop is it maintains a moving or discounted average of square of gradients. It divides gradients by the root of this average.

Have compiled the discriminator network using the rmsprop optimizer. For loss have specified binary crossentropy.

CHAPTER 5: ITERATIVE RESULTS & FINAL IMPLEMENTATION

5.1 CNN MODEL ITERATIONS AND RESULTS

In the preceding chapter we saw in detail the CNN architecture and now complied the model using the following code.

```
compile(loss = 'categorical_crossentropy',
         optimizer = 'adam',
         metrics = c('accuracy'))
summary(model)
```

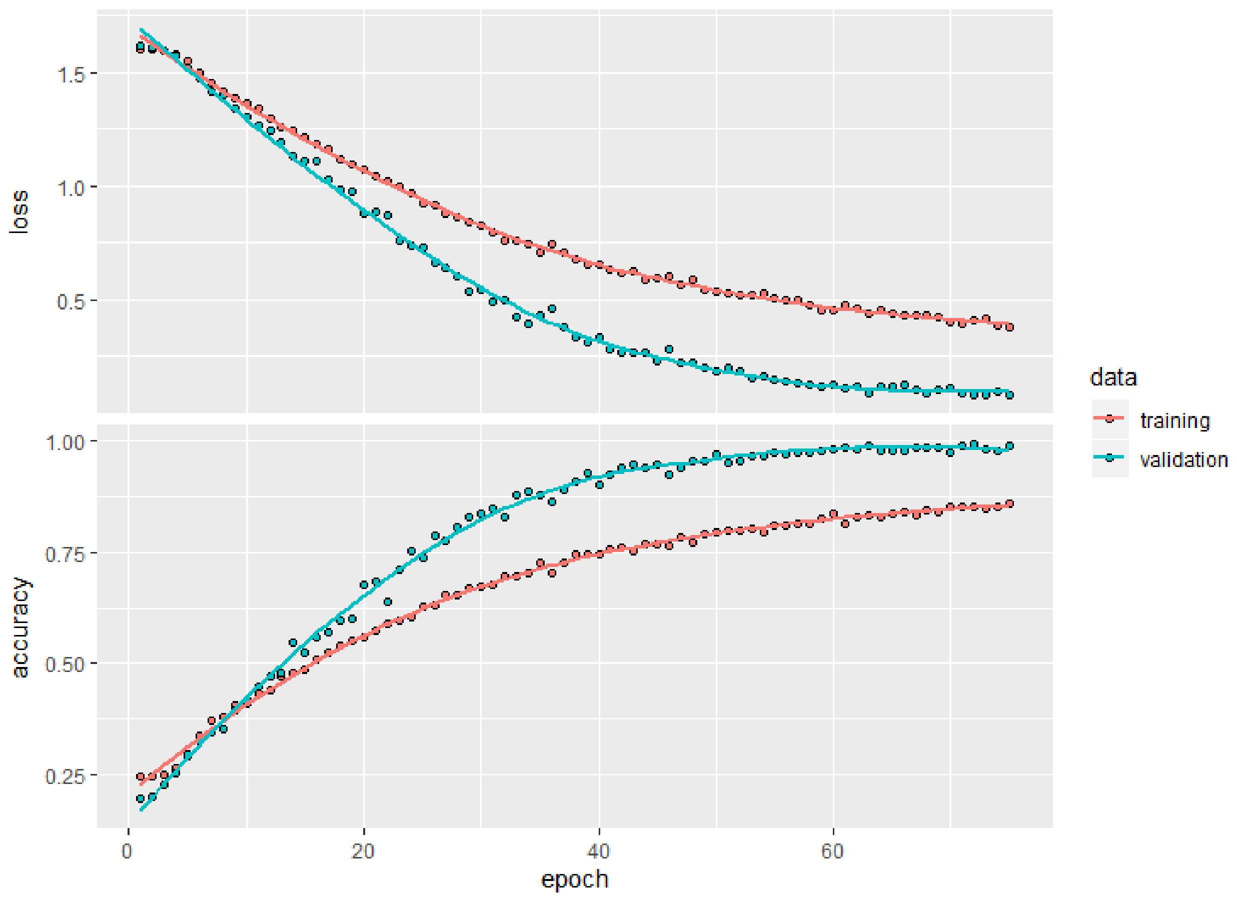
Reason for choosing the Adam optimizer is explained in chapter 4. Loss is specified as categorical crossentropy as there are 5 labels or categories of Diabetic retinopathy stages. Had iterated the model using Adadelta and RMSprop during the process of coding and results and found that accuracy of the model was higher in case of Adam optimizer and in final model used adam optimizer. Adadelta is an adaptive learning rate method for gradient descent. As the name itself suggest it does not require manual tuning at the stage of learning and dynamically adapts over the time.

Following code was used to fit the model.

```
setwd("D:/Masters in Data Analytics - LJMU/Data set/DR Dataset/Final Codings")
history <- model %>%
  fit(Train_fin_x,
      TrainLabels,
      epochs = 75,
      batch_size = 32,
      validation_data = list(Test_fin_x, TestLabels),
      callbacks = callback_tensorboard('ctg_final/one'))
```

Here used 75 epochs or iterations with batch size of 32 and used test data for validation. With additional layers of neural network each run took relatively more time but since the parameters of image was reduced from 512*512*3 to 28*28*3 the computational time was relatively shortened.

After fitting the model, the accuracy and loss values for the 75 epochs is plotted as follows.



We can observe from the preceding plot that validation accuracy continues to increase, whereas training accuracy is more or less flat. A similar trend in opposite direction is observed for the loss values. Did not observe any major overfitting problem.

Model Evaluation and Prediction

After fitting the model have evaluated its performance in terms of loss and accuracy. Also have created confusion matrix to assess classification performance across the 5 labels of Diabetic retinopathy severity. Model evaluation and prediction for both test and train data have been obtained. Also, the performance on independent data set out of test and train data has been done to evaluate the model efficacy.

Loss and accuracy based on training data are obtained as shown in following code on Train data.

```
model %>% evaluate(Train_fin_x, TrainLabels)

#=] - 1s 178us/sample - loss: 0.0452 - accuracy: 0.9893
#$loss
#[1] 0.07921534

#$accuracy
#[1] 0.9893354
```

Loss obtained is 0.079 and Accuracy of the model is 0.989 for training data.

Following is the confusion matrix based on the predicted and actual values for training data.

```
pred <- model %>% predict_classes(Train_fin_x)
table(Predicted = pred, Actual = Trainy)

#              Actual
#Predicted    0     1     2    3    4
#         0 1295    8    21    0    0
#         1    1 1032     0    0    1
#         2    5    5  1023    1    0
#         3    0    3     4  946    0
#         4    0    2     2    3  899
```

From Preceding confusion matrix following are the observations.

- The correct classifications are shown on the diagonal for all 5 labels of DR severity.
- Best classification is seen for category 4 (proliferative DR category) where 899 images are correctly classified out of 900
- Highest confusion in category of 2 (moderate DR category) where 18 images are wrongly classified.

The prediction probabilities for the sample images out of 5251 images are show below:

						Predicted_ class	Actual
1	0.769887	0.014641	0.213133	0.000755	0.001584	0	0
2	0.965755	0.004424	0.029545	6.39E-06	0.00027	0	0
3	0.982962	0.000641	0.016373	1.74E-05	6.49E-06	0	0
4	0.756191	0.081381	0.144975	0.001595	0.015858	0	0
5	0.997477	0.000473	0.001877	1.34E-08	0.000173	0	0

Some of the prediction probabilities for sample images which have been misclassified.

						Predicted_class	Actual
2385	0.731649	0.001446	0.266244	0.000236	0.000426	0	2
2402	0.542935	0.00821	0.448289	6.49E-05	0.000501	0	2
2499	0.499875	0.014163	0.479618	0.006326	1.74E-05	0	2
2531	0.536566	1.42E-05	0.450824	0.010793	0.001804	0	2
2576	0.55672	0.066141	0.284107	0.09192	0.001111	0	2
2582	0.595791	0.046728	0.343303	0.012457	0.001721	0	2
2613	0.041184	0.009614	0.305521	0.643097	0.000583	3	2
2633	0.547288	0.003734	0.406081	0.038329	0.004568	0	2
2659	0.502052	0.026258	0.418383	0.051498	0.00181	0	2
2673	0.154668	0.089771	0.31789	0.007595	0.430076	4	2

Test data

Loss and accuracy values to Test data are as follows:

```
model %>% evaluate(Test_fin_x, TestLabels)


#=] - 0s 186us/sample - loss: 0.0478 - accuracy: 0.9880
#$loss
#[1] 0.08198654

#$accuracy
#[1] 0.988
```

We observe Loss is marginally lower and lower accuracy than the Training data. Confusion matrix based on predicted and actual values are as follows.

```
pred1 <- model %>% predict_classes(Test_fin_x)
table(Predicted = pred1, Actual = Testy)
#              Actual
#Predicted   0   1   2   3   4
#          0  98   0   1   0   0
#          1   0  97   0   0   0
#          2   2   3  99   0   0
#          3   0   0   0 100   0
#          4   0   0   0   0 100
```

Following are the observations from the confusion matrix

- The model is most confused in terms of label 1 (mild DR) classification of diabetic retinopathy.
- The best image recognition performance is for label 3 (severe DR) and label 4 (proliferative DR). Overall the results are same as that of training data.

Looking at the prediction probabilities for the first five images of the test data is are correct.

						Predicted_class	Actual
1	0.99702	0.000863	0.002096	5.89E-06	1.51E-05	0	0
2	0.99834	0.001345	0.000314	1.45E-10	1.01E-07	0	0
3	0.851837	0.014225	0.132119	5.45E-05	0.001765	0	0
4	0.977852	0.004835	0.01731	2.67E-06	1.29E-06	0	0
5	0.997915	0.0019	0.00018	2.65E-06	1.56E-06	0	0
6	0.727132	0.263088	0.005511	0.003175	0.001094	0	0

With sufficiently high classification performances with both training and test data in terms of accuracy, will apply the model on independent retina images. Total of 250 images was selected along with 50 images belonging to each label of DR from 0 ,1,2,3,4 classified as No DR, mild DR, moderate DR, severe DR and proliferative DR respectively.

```
model %>% evaluate(Ind_fin_x, IndLabels)
#=] - 0s 197us/sample - loss: 2.8364 - accuracy: 0.3880
#$loss
#[1] 2.50051

#$accuracy
#[1] 0.388
```

We find that accuracy of the model drops to 38.8% and loss function increasing to 2.50. Accuracy dropped down from 98.8% to 38.8%. Hence model cannot be generalized.

To explore improvements to model image modification was done. Probably the images selected is pointing in one direction. That is more of left eye or right eye and this may be biasing the model and hence lower accuracy on independent data set.

To address this, 250 independent images were modified with a flop function that will make the images reverse in opposite from its original position.

On changing the orientation of the images, the accuracy further drops to 26.8%

```
model %>% evaluate(Ind_fin_x, IndLabels)
#=] - 0s 200us/sample - loss: 3.1559 - accuracy: 0.2680
#$loss
#[1] 3.156368

#$accuracy
#[1] 0.268
```

It can be concluded that image orientation in the train data and test data is not biased and probably more images need to be fed into CNN layers for more robust and generalized model.

5.2 SVM MODEL ITERATIONS AND RESULTS

Before we decipher the results of SVM iterations let us first understand the basics of Confusion Matrix.

	Observed/ Actual/Reference values		
		Positive (Event)	Negative (No Event)
Predicted values	Positive (Event)	A (True Positive)	B (False Positive)
	Negative (No Event)	C (False Negative)	D (True Negative)

Accuracy is calculated as the number of all correct predictions i.e. true positives and true negatives, divided by the total number of reference values in the dataset. The best accuracy is 1 and worst accuracy is 0. [19] In above matrix it can be calculated as (A+D/A+B+C+D)

Sensitivity is calculated as number of correct positive predictions divided by total number of positive events in the dataset. It is also known as recall or true positive rate. The best sensitivity one achieves is 1 and worst sensitivity is 0. [19]. In above matrix it can be mathematically represented as (A/A+C)

Specificity is calculated as number of correct negative predictions i.e. true negatives divided by total number of negative events in the dataset. It is also known as true negative rate. The best 1 and worst is 0 in terms of specificity rate. [19] In the above matrix it can be mathematically represented as (D/D+B)

Precision or positive predictive value is calculated as the number of correct positive predictions i.e. true positives divided by the total number of positive predictions, which is sum of true positives and false positives. Best precision metric is 1 and worst precision metric is 0 [19] In the above matrix it can be mathematically represented as (A/A+B)

Negative predictive value is calculated as the number of correct negative predictions i.e true negatives divided by the total number of negative predictions, which is true negatives and false negatives. Best negative predictive is 1 and Worst negative predictive value is 0. In the above matrix it can be mathematically represented as (D/C+D)

Prevalence is calculated as (A+C/A+B+C+D). It can be interpreted as how often the correct classification occurs in the data set.

Detection rate is calculated as (A/A+B+C+D).

Detection Prevalence rate is calculated as (A+B/A+B+C+D)

Balance Accuracy is calculated as (Sensitivity + Specificity/2) of [(A/A+C) +(D/D+B)]/2

Kappa Statistic compares the accuracy of the system to the accuracy of a random system. It is mathematically represented as [total accuracy – random accuracy]/[1 – random accuracy]. Where Total accuracy is (A+D/A+B+C+D) and random accuracy is expressed in terms of [{(B+D)*(C+D)+(A+C)*(A+B)}/{A+B+C+D}^2]

Correlation Matrix with 2352 independent variables was calculated to know about multicollinearity among independent variables. Following is a sample of correlation matrix of first ten variables. We observe 1 and the correlation among diagonal elements which explains that an independent variable is perfectly correlated with itself.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	1	-0.10276	-0.01768	-0.00299	0.007654	0.010546	0.007713	-0.02332	-0.16439	0.104608
V2	-0.10276	1	0.048774	-0.01032	0.013829	0.013333	0.008284	0.004916	0.102874	-0.18639
V3	-0.01768	0.048774	1	-0.10626	0.009555	0.02777	-0.09226	0.68883	0.062585	-0.04662
V4	-0.00299	-0.01032	-0.10626	1	-0.2611	-0.24772	0.673693	0.026964	0.025039	-0.00743
V5	0.007654	0.013829	0.009555	-0.2611	1	0.804892	-0.19169	0.049782	0.027584	0.002665
V6	0.010546	0.013333	0.02777	-0.24772	0.804892	1	-0.26943	0.023041	0.016854	0.00502
V7	0.007713	0.008284	-0.09226	0.673693	-0.19169	-0.26943	1	-0.19459	-0.01734	-0.00805
V8	-0.02332	0.004916	0.68883	0.026964	0.049782	0.023041	-0.19459	1	0.168205	-0.03809
V9	-0.16439	0.102874	0.062585	0.025039	0.027584	0.016854	-0.01734	0.168205	1	-0.12081
V10	0.104608	-0.18639	-0.04662	-0.00743	0.002665	0.00502	-0.00805	-0.03809	-0.12081	1

Principle component analysis was done to reduce the variables. It is a technique used to emphasize variation, bring out strong patterns and make data easy to explore and visualize. [20] Rationale behind the PCA is to reduce dimensionality in data set consisting of many independent variables which are correlated to each other while retaining the variation.

However, PCA did not reduce the variables and after PCA was done also the variables remained at same 2352 independent variables. Hence model was developed on Original dataset abandoning the PCA.

Following was the performance of SVM on Train data

```
caret::confusionMatrix(Eval_RBF,Train_df$DR_class)

#Confusion Matrix and Statistics as per results in R

#        Reference
#Prediction     0     1     2      3    4
#           0 1273   62    74     16   10
#           1      3  935   11      4   26
#           2      1    20  934      1    5
#           3      6    11     8   916   11
#           4     18    22    23     13  848

#Overall Statistics

#Accuracy : 0.9343
#95% CI : (0.9273, 0.9409)
#No Information Rate : 0.2478
#P-Value [Acc > NIR] : < 2.2e-16

#Kappa : 0.9174

#Mcnemar's Test P-Value : < 2.2e-16

#Statistics by category:

#                          Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
#Sensitivity                 0.9785   0.8905   0.8895   0.9642   0.9422
#Specificity                 0.9590   0.9895   0.9936   0.9916   0.9825
#Pos Pred Value              0.8871   0.9551   0.9719   0.9622   0.9177
#Neg Pred Value              0.9927   0.9731   0.9730   0.9921   0.9880
#Prevalence                  0.2478   0.2000   0.2000   0.1809   0.1714
#Detection Rate              0.2424   0.1781   0.1779   0.1744   0.1615
#Detection Prevalence        0.2733   0.1864   0.1830   0.1813   0.1760
#Balanced Accuracy           0.9687   0.9400   0.9415   0.9779   0.9624
```

Overall accuracy of the model was at 93.43%. Sensitivity was ranging from 88.95% to 97.85% across label of DR and Specificity was ranging from 95.90% to 99.36% across varied class category of DR.

On observation of confusion matrix following were the conclusions.

- The model is most confused in terms of label 1 (mild DR) and label 2 (moderate DR) classification of diabetic retinopathy.

- The best image recognition performance is for label 0 (No DR) and label 3 (Severe DR).

Following was the performance of SVM on Test data

```
confusionMatrix(Eval_RBF_test,Test_df$DR_class)


#Confusion Matrix and Statistics

#        Reference
#Prediction    0   1  2   3  4
#          0   97   3  10  3  0
#          1    1  86   2  0  8
#          2    0   3 82   0  1
#          3    0   0  4  95  0
#          4    2   8  2   2 91

#Overall Statistics

#Accuracy : 0.902
#95% CI : (0.8725, 0.9266)
#No Information Rate : 0.2
#P-Value [Acc > NIR] : < 2.2e-16

#Kappa : 0.8775

#Mcnemar's Test P-Value : NA

#Statistics by Class:

#                       Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
#Sensitivity              0.9700   0.8600   0.8200   0.9500   0.9100
#Specificity              0.9600   0.9725   0.9900   0.9900   0.9650
#Pos Pred Value           0.8584   0.8866   0.9535   0.9596   0.8667
#Neg Pred Value           0.9922   0.9653   0.9565   0.9875   0.9772
#Prevalence               0.2000   0.2000   0.2000   0.2000   0.2000
#Detection Rate           0.1940   0.1720   0.1640   0.1900   0.1820
#Detection Prevalence     0.2260   0.1940   0.1720   0.1980   0.2100
#Balanced Accuracy        0.9650   0.9163   0.9050   0.9700   0.9375
```

Overall accuracy of the model was at 90.2%. Sensitivity was ranging from 82% to 97% across label of DR and Specificity was ranging from 96% to 99% across varied class category of DR. The above performance statistics were little lower compared to training data.

On observation of confusion matrix following were the conclusions.

- The model is most confused in terms of label 1 (mild DR) and label 2 (moderate DR) classification of diabetic retinopathy.

- The best image recognition performance is for label 0 (No DR) and label 3 (Severe DR). The trends are same as that of train data.

We observe that sufficiently high classification performances with both training and test data in terms of accuracy for SVM modelling. Will apply the model on independent retina images data set. Total of 250 images was selected along with 50 images belonging to each label of DR from 0 ,1,2,3,4 classified as No DR, mild DR, moderate DR, severe DR and proliferative DR respectively.

The performance on model on independent data set is as follows

```
confusionMatrix(Eval_RBF_Ind,Ind_df$DR_class)

#Confusion Matrix and Statistics

#          Reference
#Prediction   0    1    2    3    4
#           0  34    8   20    7    2
#           1   5   27    5  18    0
#           2   3    4   14    5    4
#           3   4    2    3  10    6
#           4   4    9    8  10   38

#Overall Statistics

#               Accuracy : 0.492
#                 95% CI : (0.4284, 0.5557)
#    No Information Rate : 0.2
#    P-Value [Acc > NIR] : < 2.2e-16

#                  Kappa : 0.365

# Mcnemar's Test P-Value : 2.086e-05

#Statistics by Class:

#                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
#Sensitivity             0.6800   0.5400   0.2800   0.2000   0.7600
#Specificity             0.8150   0.8600   0.9200   0.9250   0.8450
#Pos Pred Value          0.4789   0.4909   0.4667   0.4000   0.5507
#Neg Pred Value          0.9106   0.8821   0.8364   0.8222   0.9337
#Prevalence              0.2000   0.2000   0.2000   0.2000   0.2000
#Detection Rate          0.1360   0.1080   0.0560   0.0400   0.1520
#Detection Prevalence    0.2840   0.2200   0.1200   0.1000   0.2760
#Balanced Accuracy       0.7475   0.7000   0.6000   0.5625   0.8025
```

Overall accuracy of the model was at 49.2%. Sensitivity was ranging from 20% to 78% across label of DR and Specificity was ranging from 81.5% to 92.5% across varied class category of DR.

On observation of confusion matrix following were the conclusions.

- The model is most confused in terms of label 2 (moderate DR) and label 3(severe DR) classification of diabetic retinopathy.
- The best image recognition performance is for label 4 (proliferative DR) and label 0 (No DR).

We find that accuracy of the model drops to 49.2% from that of 93.43% and 90.2% from train and test data sets respectively. Model is a stable one and to generalize further images need to be added to data set for effective learning.

Other observation was that SVM model is more stable than that on CNN model on independent data set as the SVM per say requires lesser data than that of neural networks.

5.3 GAN MODEL ITERATIONS AND RESULTS

We have seen the architecture of generator and discriminator networks in chapter 4 used for modelling. Finally, have designed a GAN which links the generator and discriminator.
On training this model will move the generator in a direction that improves its ability to deceive the discriminator. So, on training the weights of generator model is updated in a way that makes discriminator more likely to predict as real images when it is looking at a fake or unreal image.

Following is the adversarial or GAN network set up

```
# adversarial network
freeze_weights(d)
gani <- layer_input(shape = l)
gano <- d(g(gani))
gan <- keras_model(gani, gano)
gan %>% compile(loss = "binary_crossentropy",
            optimizer = optimizer_rmsprop(lr = 0.0004,clipvalue = 1.0,decay = 1e-8))
```

Adversarial network sets discriminator weights to non-trainable. For each epoch or iteration following was performed by the GAN model network

- Generates random points (random noise) in the latent space
- Creates new images with generator using this random noise
- Mixes the generated unreal images with real images
- Train the discriminator network using the mix of real and unreal images with corresponding targets; either “real” for real images or “fake” for unreal images.

- Draw new random points in the latent space
- Train GAN using these random vectors, with targets that say, “these are all real images”. This update the weights of the generator to move towards discriminator to predict “these are real images”. This trains the generator to deceive the discriminator.
- To know the efficacy of adversarial and discriminator network metrics of a_loss (adversarial loss) and d_loss (discriminator loss) was used.

On First iteration saw that adversarial loss was 10.95594 and discriminator loss was -0.81200. Discriminator was ending up dominating the Generator. To counter this discriminator model architecture was changed to reduce the learning rate and increase the dropout rate.

On removing 2 dense layers and reducing the dropout rate for 0.4 to 0.25 in discriminator model the final adversarial loss was 1.33927 and discriminator loss was 0.57423

A consistent Generative Adversarial Network will have a discriminator loss around 0.5, typically in range of 0.5 and maybe high as 0.7 or 0.8. The generator loss will be higher than discriminator loss and ranges around 1.0, 1.5, 2.0, or slightly higher.

Based on above benchmark have stabilised our GAN as generator loss and discriminator loss is well within the consistent range.

To wrap up Generator never sees the image from training data and image stream comes from the generator network out of random points in latent space. GAN’s are difficult to train because as it is a dynamic and live process rather a mere gradient descent process. Extensive tuning of discriminator and generator models is required so that generator model or discriminator model does not dominate each other.

Since GAN’s learn through a dynamic process certain practicality of image editing through latent space vector is not possible.

CHAPTER 6: CONCLUSIONS

6.1 COMPARATIVE RESULTS OF ALL MODELS ON DR PREDICTION

	Train Data		Test Data		
Parameters	Convolutional Neural Networks	Support Vector Machines	Convolutional Neural Networks	Support Vector Machines	Generative Adversarial Network
Loss Function	0.0792		0.0819		
Accuracy	0.9893	0.9343	0.9880	0.9020	
Kappa		0.9174		0.8775	
P-Value		2.20E-16		2.20E-16	
Adversarial Loss					1.3393
Discriminator Loss					0.5742

6.2 CONCLUSIONS

The aim of this study was to arrive at which back propagation models between CNN, SVM and GAN were better for prediction of diabetic retinopathy using ocular images. Objective of the study was achieved through this research and following are the conclusion of this dissertation

- CNN seems to have higher accuracy than SVM in terms of DR prediction both on train and test data

- SVM model performs better and is stable on independent images than compared to CNN model. We saw in chapter 5 that SVM returned an accuracy of 49.2% as against 38.8% on independent data set of 250 DR images.
- GAN also performed very well in terms of returning within benchmark band of parameters for adversarial loss and discriminator loss with 1.3393 and 0.5742 respectively.
- CNN requires more images for learning than SVM to have stable model for prediction on independent data set

6.3 LIMITATIONS OF RESEARCH

Following are the limitations encountered while doing this research

- Could have used Graphical Processing Units (GPU) thus enabling higher computational power and ensuring more images were fed to neural networks, support vector machines and Generative Adversarial Nets.
- More images learning would have given better accuracy rates for SVM and CNN and would have created a more stable model while applying the same on Independent data set.
- To optimise on computational speed and time reduced the initial images of 512*512*3 to 28*28*3. If GPU environment was available than could have reduce the images to better resolution of 100*100*3 and thus more features and dimensions of images would have been included in the algorithm resulting in increased accuracy and stability of the model.

- Stability and usefulness of GAN in prediction of class of images could have been further explored as part of the study

6.4 FUTURE RESEARCH OPPORTUNITY

Following are the research opportunity available in this area of study

- Creating an Ensemble model for prediction by combining one or more back propagation algorithms
- Using Informational GAN for prediction of class labels or categories of diabetic retinopathy severity amongst the ocular images
- Developing an application-based utility where images can be scanned from a mobile or image processing hardware and returning the prediction of retina in terms of Diabetic retinopathy severity.
- Develop a model each one for Left and right eye, so that model accuracy and stability further improves

6.5 CONTRIBUTIONS TO THE RESEARCH

Following are the contributions available in this area of study

- Study found that SVM requires less data for training and will result in better stable model than CNN.
- A code in R which gives Prediction probabilities for severity of diabetic retinopathy
- Application of GAN for first time in Diabetic retinopathy prediction

REFERENCES

[1] www.allaboutvision.com

[2] www.medicalnewstoday.com/articles/183417.php#symptoms

[3] Automatic Recovery of the Optic Nervehead Geometry in Optical Coherence Tomography Kim L. Boyer* , *Fellow, IEEE,* Artemas Herzog, *Student Member, IEEE,* and Cynthia Roberts , IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 25, NO. 5, MAY 2006

[4] Novel Fractal Feature-Based Multiclass Glaucoma Detection and Progression Prediction, Paul Y. Kim, Khan M. Iftekharuddin, *Senior Member, IEEE*, Pinakin G. Davey, M´arta T´oth, Anita Garas,Gabor Holl´o, and Edward A. Essock, IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, VOL. 17, NO. 2, MARCH 2013

[5] Optic Disc Localization using Local Vessel Based Features and Support Vector Machine Anum Abdul Salam, M. Usman Akram, Member, IEEE, Sarmad Abbas, Syed M. Anwar

[6] Prediction of Glaucoma through Convolutional Neural Networks , Marco A. Espinoza1, Germán H. Alférez1, and Javier Castillo2 School of Engineering and Technology1, Instituto de la Visión del Hospital La Carlota2 Universidad de Montemorelos Apartado 16-5 Montemorelos, N.L. Mexico E-mails: 1120631@alumno.um.edu.mx1, harveyalferez@um.edu.mx1, oftalmo@um.edu.mx2 ,Int'l Conf. Health Informatics and Medical Systems | HIMS'18

[7] https://skymind.ai/wiki/generative-adversarial-network-gan, A Beginner's Guide to Generative Adversarial Networks (GANs)

[8]https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29 , Understanding Generative Adversarial Networks (GANs) Building, step by step, the reasoning that leads to GANs. Joseph Rocca

[9] Pathological Evidence Exploration in Deep Retinal Image Diagnosis
Yuhao Niu, Lin Gu, Feng Lu, y Feifan Lv, Zongji Wang, Imari Sato, Zijian Zhang, Yangyan Xiao, Xunzhang Dai, Tingting Cheng

[10] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way – Sumit Saha

[11] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ Understanding Support Vector Machine algorithm from examples, Sunil Ray, Sept 13, 2017.

[12] http://www.statsoft.com/textbook/support-vector-machines

[13] https://machinelearningmastery.com/support-vector-machines-for-machine-learning/

[14] https://www.bioconductor.org/packages/release/bioc/vignettes/EBImage/inst/doc/EBImage-introduction.html

[15] https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f - Anish Singh Walia

[16] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/RMSprop

[17] Advanced Deep Learning with R – Bharatendra Rai, Packt Publications

[18] Deep Learning with R – Francois Chollet with J J Allaire, Manning Publications

[19] https://classeval.wordpress.com/introduction/basic-evaluation-measures/

[20] http://setosa.io/ev/principal-component-analysis/ - By Victor Powell

[21] https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/ - Jason Brownie

[22] https://www.kaggle.com/c/diabetic-retinopathy-detection/data

[23] https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/

[24] https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

[25] Deep Learning (Adaptive Computation and Machine Learning series) – Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach – The MIT Press

[26] Advanced Deep Learning with Keras – Rowel Atienza – Packt Publications

[27] Generative Deep Learning – David Foster – O'Reilly Publications

[28] Statistics for Data Science – James D Miller – Packt Publications

[29] Practical Statistics for Data Scientists – Peter Bruce and Andrew Bruce – O'Reilly Publications

[30] Tensor Flow Machine Learning Projects – Ankit Jain, Armando Fandango & Amita Kapoor – Packt Publications

APPENDIX 1

[App 1] Link of detailed R code in Github

https://github.com/vikram-sreedhar/R-Codes-for-DR-Prediction/blob/master/Final_modeling_dr_Itr2.R