

A Novel Optical Coherence Tomography Scan Based Diabetic Retinopathy Detection using Neural Network

Major project report submitted in partial fulfilment of the requirement for the degree of
Bachelor of Technology

in

Computer Science and Engineering

By

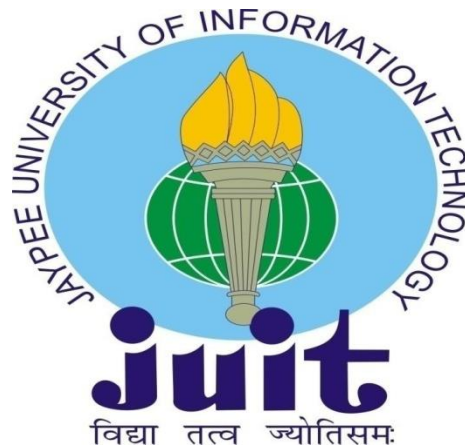
Akshit Aggarwal (171294)

Akarsh Gupta (171277)

UNDER THE SUPERVISION OF

Dr. Amol Vasudeva

to



Department of Computer Science & Engineering and Information Technology

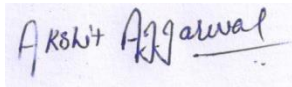
Jaypee University of Information Technology Wagnaghat, Solan 173234,

Himachal Pradesh, INDIA

I
CERTIFICATE

Candidate's Declaration

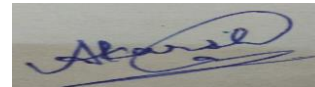
I hereby declare that the work presented in this report entitled “**A Novel Optical Coherence Tomography Scan Based Diabetic Retinopathy Detection using Neural Network**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Wanknaghat is an authentic record of my own work carried out over a period from January 2021 to June 2021 under the supervision of **Dr. Amol Vasudeva (Assistant Professor, Computer Science & Engineering and Information Technology)**). The matter embodied in the report has not been submitted for the award of any other degree or diploma.



<Signature>

Akshit Aggarwal

171294

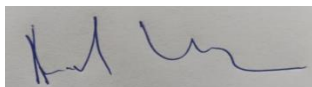


<Signature>

Akarsh Gupta

171277

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.



<Signature of the Supervisor>

Dr. Amol Vasudeva

Assistant Professor

(Computer Science & Engineering and Information Technology)

Dated: 17-06-2021

Head of the Department/Project Coordinator

II

ACKNOWLEDGEMENTS

Thanks to invincible creator who grant us grit and mastery to begin and later on complete this exertion on project, gratitude to **HOD** Sir and our mentor **Dr. Amol Vasudeva**, who usher and assist us at the distinct stages of the project.

We would also like to gratitude my panel **Dr. Rajinder Sandhu** and **Dr. Mrityunjay Singh**, for providing us a chance to represent my project and for investigating my exertion and giving me the acknowledgement which would surely assist us in the future.

Finally yet importantly, we would like to gratitude our institution **Jaypee University of Information Technology** for giving me a tenet to carry out and showing my skills of the various fields which I have studied till date.

III
TABLE OF CONTENTS

Chapter		Page
1.	Introduction	1-4
	1.1 Introduction	
	1.2 Symptoms	
	1.3 Objectives	
	1.4 Problem Statement	
	1.5 Methodology	
	1.6 Scope	
2.	Literature Survey	5-7
	2.1 Literature Review	
	2.1 Our Contribution	
	2.2 Literature Conclusion	
3.	System Development	8-22
	3.1 Background	
	3.2 Overview of Model	
	3.3 Data Collection and Preprocessing	
	3.4 Feature Selection and Extraction	
	3.5 Classification	
4.	Performance Analysis	23-40
	4.1 System Requirement	
	4.2 Library Requirement	
	4.3 System Development Approach	
	4.4 Installation of Packages	
	4.5 Discussion of Results and Comparison	
5.	Conclusion	41

5.1 Advantages

5.2 Conclusion

5.3 Future Work

6.	References	42-44
	Appendices	45

IV

LIST OF ABBREVIATIONS

Adam	Adaptive Moment Estimation
RGB	Red Green Blue
CNN	Convolutional Neural Network
Conv2d	Convolutional 2-Dimensional
DR	Diabetic Retinopathy
DRuNN	Diabetic Retinopathy using Neural Network
GUI	Graphic User Interface
NPDR	Non-Proliferative Diabetic Retinopathy
OCT	Optical Coherence Tomography
PDR	Proliferative Diabetic Retinopathy
ReLU	Rectified Linear Unit
RMS	Root Mean Square
ROC	Receiver Operating Characteristic
VGG	Visual Geometry Group

IV

LIST OF SYMBOLS

a_n	Input Image
m_n	Weight Corresponds to each layer
I	Input Layer
H	Hidden Layer
$f(a)$	Activation Function
B	Biassing Index
O_n	Output Corresponds to Input Image
a^d	Vector on Input Layer
m^d	Vector Corresponds to Weight
B^d	Biassing Vector
O^d	Output Vector
W	Polling Window Size
S	Stride
a_x	Dimension of Input Image along Horizontal Direction
a_y	Dimension of Input Image along Vertical Direction
γ	Kernel
p_i	Input pixel
o_i	Output image pixel
p_x	Pixel row
γ_x	Kernel row
Z	Padding
N	Width of Convolutional filter
R	Height of Convolutional filter

V

LIST OF FIGURES

Fig #	Topic	Page
1	Human Eye	1
2	Diabetic Retinopathy Affected Patient's OCT Scan	1
3	Normal Retina Patient's OCT Scan	2
4	Blurred Vision, Dark Vision OCT Scan	2
5	Problem Statement (Normal or Affected Retina)	3
6	Overview of Methodology	4
7	Flowchart of Proposed Framework	10
8	Overview of Data Collection and Scaling of Pixels into 1-D array	11
9	Convolutional Neural Network Model	12
10	Operations for generating Feature Map	13
11	Layers in CNN	14
12	Input Image in Tensor Form	14
13	Average Pooling	15
14	Feature Map	17
15	ReLU Model Architecture	19
16	Working of Visual Geometry Group-16 Model	19
17	Working of Adam Optimizer Reducing Error	22
18	NumPy Library Working	23
19	Pandas Library Internal Working	24
20	Flow of SkLearn Library	25
21	Tensor in Array Format	26
22	Flow Diagram of Tensorflow Library	26
23	Life Cycle of Neural Network in Keras	27

24	Working of OpenCV Library	28
25	GUI Based Window	28
26	Image Classification	29
27	Sequence Diagram	29
28	Use Case Diagram	30
29	Class Diagram	31
30	Activity Diagram of DRuNN Technique	32
31	Data Flow Diagram	33
32	Installation of Packages	34
33	Comparison of Accuracy with other existing Imagenet models	37
34	Comparison of Performance Parameters Value	38
35	Output Window of GUI screen of Normal Scan	39
36	Output Window of GUI screen of DR affected Scan	39

VI
LIST OF GRAPHS

Graph #	Topic	Page
1	Simple Plot by Matplot Library	24
2	Attractive Graph by Seaborn Library	25
3	ROC Curve	40

VII
LIST OF TABLES

Table#	Topic	Page
1	Comparison of Existing DR Detection Technique	6-7
2	VGG-16 Model with training Parameters	19-20
3	Simulation Parameters	35-36
4	VGG-16 Model with variation in Training dataset values	37-38

VIII

ABSTRACT

Diabetic Retinopathy (*DR*) is a burgeoning malady in Asian territories. *DR* causes 5% to 7% of the total blindness throughout the region. In India, the record of *DR* affected patient will reach around 79.4 million by 2030. The main aim of this research is to determine whether a patient is agonizing from *DR* or not by the dint of Optical coherence tomography (*OCT*) scans. In this regard, Engineering based technique such as deep learning and neural network play methodical role to fight against this fatal disease. So, in this proposed research work, a model based machine using Deep Learning and Neural Network, is generated. This machine examines the feature of *OCT* scan of *DR* affected eye and normal human eye. With the support of Convolutional Neural Network (*CNN*) (part of Neural Network) and with its layers like *Conv2D*, *Pooling*, *Dense*, *Flatten*, *Dropout* the model helps to understand the curve and color based feature of the scan. The Visual Geometry Group (*VGG-16*) model and Adaptive Moment Estimation optimizer is used for training and reducing error for the developed model. Further, for user suitability a *GUI* based system is developed which opens a *GUI* based window. The proposed research contribution definitively detects whether the given *OCT* scan with an efficient approach for finding *DR* affected persons within few seconds.

CHAPTER 1

(INTRODUCTION)

1.1 Introduction

Diabetic Retinopathy (*DR*) is a burgeoning malady. In India, the rate of *DR* affected patients will reach around 79.4 million by 2030 [1]. *DR* affects the human eyes. The human eye mainly consists of three layers such as lens, cornea, and retina (as shown in Fig. 1). The retina is examined as an important part of human eye because all the visual image is formed on this important part.

The main reason of *DR* is Diabetes. The diabetes affects the human kidneys, nerves, human eyes, raise the fats in the blood, and also increase the blood pressure. It increases the sugar level in body that blocks the oxygen in blood.

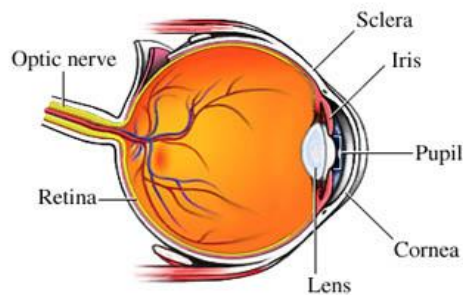


Fig. 1: Human Eye

So the human eye's retina will not get the oxygen. New blood cells are formed in the human retina due to the blockage of oxygen (as shown in Fig. 2). The Newly blood cells are not properly matured for the retina. These blood cells get damaged due to which retina produced a liquid material called as *Vegf*. The liquid material mix up with the gel which is present in the human eyes that causes blindness. This whole process is considered as Diabetic Retinopathy.

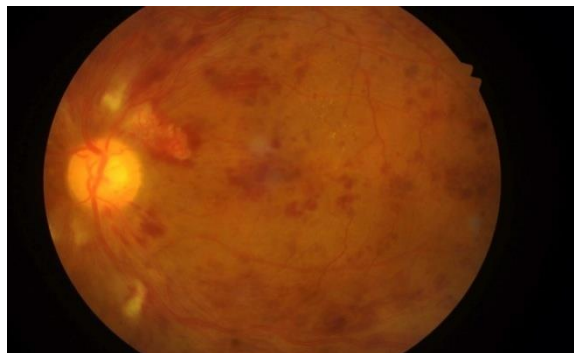


Fig. 2: Diabetic Retinopathy affected Patient's *OCT* scan



Fig. 3: Normal Retina Patient's *OCT* scan

DR is mainly split into two basic classes Non-Proliferative Diabetic Retinopathy (*NPDR*) and Proliferative Diabetic Retinopathy (*PDR*). In case of *NPDR* no new blood cells are formed which is consider as an initial stage of *DR*. *NPDR* is easy to handle. In case of *PDR*, new blood cells are formed in the retina which is consider as an onerous stage which is quite difficult to handle. Initially, *DR* shows no symptoms but later it affects the human eyes. So, the main symptoms of this disease are that a person might see spot floating in vision, blurred vision, and dark or empty area in vision. If a person is suffering from these kind of symptoms then the physician will advise for the *OCT* test. The physician will see the *OCT* scan report and on the behalf of these scan the physician will predict whether the person is suffering from *DR* or not.

This research paper proposes a *GUI* based technique namely, Diabetic Retinopathy Detection using Neural Network *DRuNN*, that helps in detecting through *OCT* scans for patients that either suffering from *DR* or not. *DRuNN* technique studies the feature of *OCT* Scan images and helps to give the prediction regarding the given *OCT* scan.

1.2 Symptoms

The most and serious indications of *DR* are a person might see spot floating in vision, blurred vision, and dark or empty area in vision.

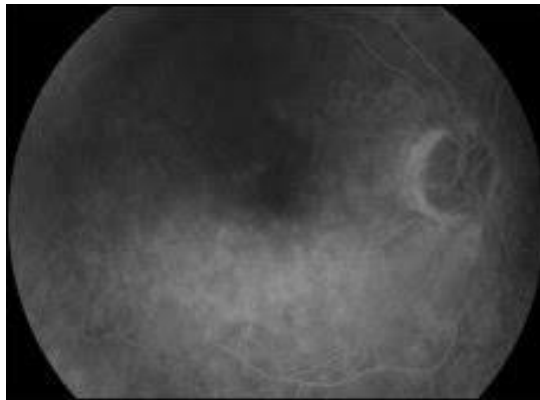


Fig. 4: Blurred Vision, dark vision *OCT* scan

1.3 Objectives

- Identifying Between the *DR* and Normal Retina.
- Step By Step Feature Learning
- Cost Reliable
- Minimum Complexity
- Making *DRuNN* most trustworthy
- Good Value of Performance Parameters.
- Should be effective for clinician purpose.

1.4 PROBLEM STATEMENT



Fig. 5: Problem Statement (Normal or *DR* affected retina)

As, *DR* is a burgeoning malady. In India, the rate of *DR* affected patients will reach around 79.4 million by 2030. So, the chief task of this study is to apply the Image processing technique on Medical chest images of x-ray scans based scan and to identify the category of X-ray scan (as shown in fig. 5).

Firstly, to solve this ambiguity of *DR* vs Normal images. Pass the images to filtration for removing noise and then check is it working or not and in second step apply image enhancement technique for study the sharpening of images means edges is retrieved or not. Afterwards, applying *CNN* algorithm and check its working or not and then apply appropriate imagenet model such as *VGG-16*, *VGG-19*, InceptionV3, Googlenet, Alexnet and compute the model in terms of performance parameter and find which model should be accepted. If the model produces some error rate then find the best suitable optimizer in case of applied *DRuNN* technique we surely apply Adam Optimizer because it is one of the best optimizer till studies and for use suitability develop a *GUI* based software.

1.5 Methodology

The *DRuNN* consists three phases namely data collection and pre-processing, feature selection, and classification (as shown in Fig. 6). The first phase, i.e., data collection and preprocessing takes the dataset of images which are collected from Messidor having 100 images. The second phase as feature selection contains *CNN* which is applied on *VGG-16* model. The final stage i.e., classification discusses for prediction on *GUI* based technique. Each phase is described.



Fig. 6: Overview of Methodology

1.6 Scope

As all the world is suffering from this epidemic disease the developed *DRuNN* technique is one of the best and available at cheaper cost technique. Since, the scope of this research based project is very wide. Many Large organization such as Google with the collaboration with All India Institute of Medical Science (Aiims), New Delhi are working on this similar projects. The proposed *DRuNN* technique is very efficient and we can consider it will be helpful for the society and also reduces the burden on doctors.

CHAPTER 2

(LITERATURE SURVEY)

2.1 Literature Review

This section highlights the most exhaustive research in the field of detecting *DR* with the help of Deep Neural Network. These research includes data collection of images, studying the feature of Images, normalizing the features, finding accuracy and efficiency. The efficiency reflects the model's development process, the speed of execution and the accuracy. The various number of techniques are presented for the efficiency such as model selection, dataset of images. Among them the model selection and accordance with the number of images plays a vital role and are discussed.

The author mainly Gulshan *et al.* have designed the development and validation of a deep learning algorithm detection of *DR* retinal Fundus [2]. The author have considered large value of dataset by attaining high value of reactivity and selectivity to recognize ascribable *DR*. The disadvantage of this design is that the applied algorithm may not perform well for images with subtle findings that a majority of ophthalmologists would not get identify. To its improvement, the author mainly Doshi *et al.* have formulated *DR* detection using Deep *CNN* model on five steps of the disease based on seriousness [3]. The disadvantage of this study is that the single model quadratic weighted kappa metric accuracy is 38.6% which is quite low. Few of the authors have extended Doshi approach by taking fundus photograph images base on the severity [4]. The limitation of Harry *et al.*'s study is the achieved accuracy is 75% which is quite low and validation images takes 188 seconds for running on *CNN* network which is quite high. To its improvement, Prentasic and Ioncaric has investigated the detection of exudates in fundus photographs using deep neural networks and anatomical landmark detection fusion [5]. The author have used a deep neural network model for detecting *DR*. The investigation achieves an accuracy as lower as 78%. The author mainly Bhatia *et al.* have tailored diagnosis of *DR* using machine learning classification [6]. The finding focus on commitment about existence of malady after bearing the machine learning algorithm on feature extracted retinal images. The main disadvantage of this study is that the inputted data is not in the form of images. A Machine Learning Ensemble Classifier (*ML-BEC*) for early prediction of *DR* technique is developed by Somasundaram and Alli [7]. The main disadvantage of this technique is that although it gives good accuracy prediction but it does not used raw data. To improvise, few of the author have postulated an identification of *DR* in eye image using transfer learning [8]. The author use Inception V3 network for the prediction of *DR* and achieved a good efficiency, highest accuracy of 48.2%. The main limitation of postulated study is that the accuracy is very low. Also, some author have extended Sarfaraz study by applying an artificial intelligence approach to disease staging namely, deep learning for improved staging of *DR* [9]. The approach uses GoogleNet model for the classification purpose of *DR* and attained an accuracy of 81%. The main limitation of this approach is that GoogLeNet model takes too much training time for dataset. The model suggested by Ting *et al.* for the classification purpose of *DR* is trained by applying large dataset of images and the accuracy is nearly comparable to those by Takahashi *et al.* [10]. The main

disadvantage of suggested model is that the vision outcomes are not properly cleared. The author *i.e.*, Gupta and Chhikara have designed the *DR: Present and Past* [11]. The author have used two steps for detecting *DR i.e.*, feature extraction and classification. The main limitation of designed framework is that there is no proper feature extraction technique and has no proper specificity of designed model. To its improvement, some author have postulated a retinal disease detection using machine learning technique [12]. The author have developed the screening system by performing classification of *DR* using three different classifier. The model achieved 82.85% accuracy but the main constraint of this development is that the data is in the form of raw part not in images form. To improves, automated detection of *DR* using neural network model has been [13]. The investigators have used Googlenet, Alexnet and other Imagenet models which improves the efficiency of model but this model provides 74.5%, 68.8% and 57.2% for Googlenet, Alexnet, and Imagenet model, respectively. The disadvantage of investigated model is that the accuracy level is low. Few researcher have have extended the Carson work by designing deep learning fundus image analysis for *DR* and macular edema grading [14]. They have designed a deep learning based system that classify the referable scan belong to the class of *DR* or not. The main disadvantage of designed study is the used dataset also contains the non dilated pupil images which is not recommended by the physicians. To its improvement, some of the authors have formulated the work by considering on classifying the seriousness of *DR* employing the five level *PIRC* scale but some features are missing in this model [15]. Further, a recent development of detection method have been studied for the diagnosis of *DR* [16]. The developed approach is intelligence based computational knowledge and processing of Images done through pixels. The model does not study the layers wise feature learning. Few author have extended the previous study by deducing computer assisted diagnosis for *DR* based on fundus image using Deep *CNN* [17]. The prepared model achieves an accuracy of 86.17%. The main limitation of this model is that the lesson of divided dataset does not contain equal image. The various cause and clinical impacts of loss to follow up in patients with *PDR* has been formulated [18]. The study determines the *PDR* and to determine that the model is suitable for clinical aspects or not. The main disadvantage of this approach is only consider the *PDR* not *NPDR*. Also, few authors have designed early detection of *DR* using *PCA-Firefly* based deep learning model [19]. The model uses firefly algorithm for implementing the dimensionality reduction but the main disadvantage of this model is that it uses the raw dataset instead of Image dataset. To its improvement, the author have deduced exudate detection for *DR* using pre-trained *CNN* [20]. The approach uses the Visual Geometry Group (*VGG-19*) model for detecting the *OCT* based scan is of *DR* affected person or not. The model gives the accuracy of approximately 95% which is quite high but the main disadvantage of this model is that *VGG-19* model increase the layers in the model so this approach increases the complexity with computing and detecting *OCT* scan based *DR*. Table 1 contains the comparison of existing *DR* detection technique.

Table 1 Comparison of existing DR detection Technique

Author(s)	Pros	Cons
Takahashi <i>et al.</i> [9]	The achieved accuracy is 81% which is high.	It increases the complexity of the model

Gupta <i>et al.</i> [11]	Uses two step procedure of extraction and classification.	There is no proper feature learning.
Lam <i>et al.</i> [13]	Uses various ImageNet model to classify the category of <i>OCT</i> scan.	The achieved accuracy is quite low.
Sahlsten <i>et al.</i> [14]	It classify the referable scan belong to the class of <i>DR</i> or not.	The variation in the collected dataset.
Qureshi <i>et al.</i> [16]	Intelligent based computation knowledge makes model reliable.	Step by step feature learning for each layer is missing.
Yeh <i>et al.</i> [17]	The achieved accuracy is 86.17% which is reliable.	Dataset is imbalanced.
Hazem <i>et al.</i> [18]	The model can be adopted for clinical persepective in a large scale.	The outcome only detect the <i>PDR</i> not <i>NPDR</i>
Mateen <i>et al.</i> [20]	The achieved accuracy is 95% which is quite high.	It increases the complexity of the model.

2.2 Our Contribution

1. The research has presented a literature review on the approaches for detecting *DR* indicating their benefits and limitations.
2. The research has initiates a *DRuNN* technique for detecting *DR* which is fully automated.
3. *DRuNN* helps in detecting *DR* through *OCT* scan with in some seconds efficiently and user do not need any kind of manual feature extraction.
4. The reliability of *DRuNN* is evaluated in terms of complexity, and value of performance parameters, and it provides minimum complexity, and reliable accuracy.
5. *DRuNN* plays an important role for estimating *DR* by a patient when they had their own *OCT* scan of eye while the physician may not have enough time to check the scan.

2.3 Literature Conclusion

Most of the techniques for the detection of *DR* using neural network based models has disadvantages of High Complexity, no feature learning for every layer, low accuracy, imbalanced dataset, and no proper *DR* stage wise detection. Therefore, there is a need to develop an efficient technique with proper feature learning which computes the result in minimum time with reliable value of performance parameters. The primary goal of this research is to proposed a technique *i.e.*, *DRuNN* which computes the result in minimum time and classify the *OCT* scan with proper *DR* stage level with high value of performance parameters.

CHAPTER 3

(SYSTEM DEVELOPMENT)

3.1 Background

This section discuss some basic terminology and background of *CNN* algorithm. Further, Adam optimizer is discussed that is used in the proposed research study.

Basic concept of *CNN*

Convolutional neural network works on the three basic concepts, namely, curve, color, and edge base filtering. This curve, color, and edge base filtering studies the curve of *OCT* scan like indicating the blood vessel, color of *OCT* scan like converting the scan into binary scale form, and edge base filtering studies the edge of *OCT* scan like hemorrhagic detachment. This filtration technique easily performs with the help of convolutional layer as discussed in Section 4.2. Afterwards, the layer generates the feature map which is helpful for predicting the category of *OCT* based scan.

Suppose $a_1, a_2, a_3, \dots, a_n$ be the input images and $m_1, m_2, m_3, \dots, m_n$ be the weights corresponds to each image. The input image is passes through the input I and hidden layers H . Each input layer and hidden layer studies the features of images such as curve, edge, and color. These layers filter the properties of image and generate an activation function, $f(a)$, with a biasing index B to maintain the non-linearity of the image. The output O_n corresponds to image is given as

$$O_n = f\left(\sum_{i=1}^n a_i x m_i + B\right) \quad (1)$$

The first layer is convolutional layer which generates the tensor of output. Suppose a_n be the input image \vec{a}_x be the vector on input image vector, and $\vec{\gamma}_x$ be the kernel vector. So the mathematical approach for conv2d layer can be represented as –

$$a_n = \vec{a}_x \cdot \vec{\gamma}_x \quad (2)$$

Further pooling layer is essential for reducing the dimensionality of image. Let, a_n be the input image, w be the pooling window size, and s be the stride. So the mathematical approach given as –

$$\left\lfloor \frac{a_n - w}{s} \right\rfloor + 1 \quad (3)$$

Afterwards, dropout layer prevents the model from overfitting. Suppose a neural network has H hidden layers $d \in \{1, \dots, H\}$. a^d be the vector of input, O^d be the vector of outputs. m^d , B^d be the weights and biases, respectively, at layer, d ; and $f(a)$ be the activation function, then it is given as –

$$a^{(d+1)} = m^{(d+1)}O^H + B^{(d+1)} \quad (4)$$

and

$$O^{(d+1)} = f(a^{(d+1)}) \quad (5)$$

Flatten layer is helpful for generating the feature map. Let a_x , a_y be the dimension of the input image, and γ denotes the pooling output so the feature map can be calculated as-

$$F = A[a_x, a_y] = (a * \gamma)[a_x, a_y] = \sum_i \sum_j \gamma[i, j]a[a_x - i, a_y - j] \quad (6)$$

Adaptive Moment Estimation (*Adam*)

The Adam optimizer is useful for reducing the error rate on developed model.

Let α be the stepsize and $\beta_1, \beta_2 \in [0, 1)$ be the exponential rates for moment estimates. $f(\theta)$ be the stochastic objective function with parameters θ . θ_0 be the initial parameter vector. g_t be the gradients, ϕ_0 be the 1st moment vector. v_0 be the 2nd moment vector, and t be the timestamp then mathematical approach can be written as-

$$g_t = \nabla_{\theta} f_t(\theta_{t-1}) \quad (7)$$

$$\phi_t = \beta_1 \cdot \phi_{t-1} + (1 - \beta_1) \cdot g_t \quad (8)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t \quad (9)$$

where $\hat{\phi}_t$, and \hat{v}_t be the updated bias first and second moment estimation

$$\hat{\phi}_t = \phi_t / (1 - \beta_1^t) \quad (10)$$

$$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t) \quad (11)$$

$$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{\phi}_t / ((\text{sqrt}(\hat{v}_t) + \epsilon)) \quad (12)$$

3.2 Overview of Method

The *DRuNN* consists three phases namely data collection and pre-processing, feature selection, and classification. The first phase, i.e., data collection and preprocessing takes the dataset of images which are collected from Messidor having 100 images. The second phase as feature selection contains *CNN* which is applied on *VGG-16* model. The final stage i.e., classification discusses for prediction on *GUI* based technique. Each phase is described.

Firstly an objective function, i.e, variation on training is applied on collected *OCT* based scans. After collection of scans feature extraction is performs on scans. Feature extraction applies the *CNN* algorithm and with the help of *CNN* algorithm, feature map is generated. Afterwards, the model is trained with transfer learning model. Further, optimizer is applied on proposed framework for reducing the error rate. After optimization algorithm, the value of performance parameters is calculated which checks the efficiency of proposed framework. Further, for user suitability a *GUI* based system is developed which selects the image, if the selected scan is of *DR* patient, it goes to the positive class, otherwise in the negative class (as shown in Fig. 7).

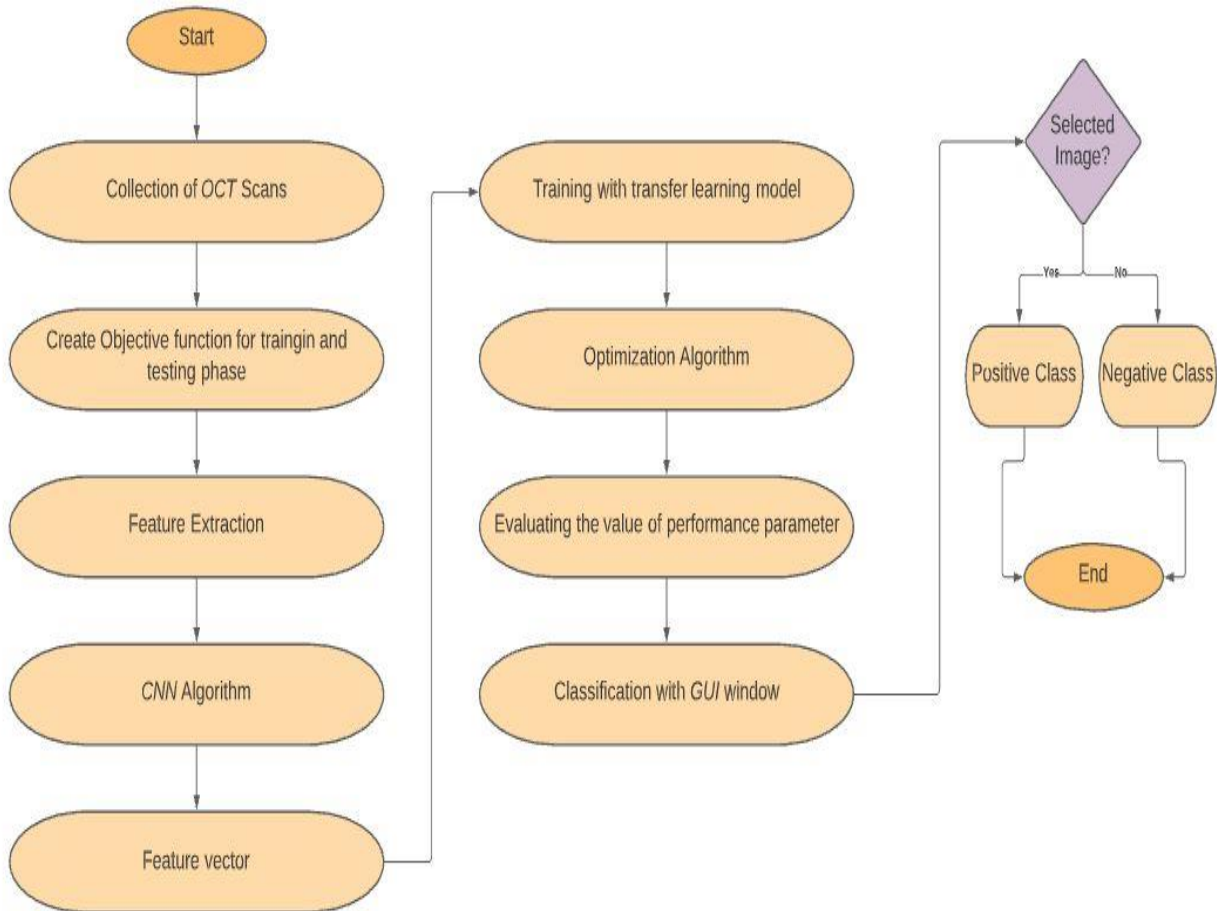


Fig. 7: Flowchart of Proposed Framework

3.3 Data Collection and Preprocessing

This is an initial phase of developing *DRuNN* technique where dataset of images are collected from Messidor (<http://www.adcis.net/en/Download-Third-Party/Messidor.html>). It consists of approximately 1200 images. For training the proposed *DRuNN* technique, 100 tricky images are randomly selected which are dilated pupil images consisting of equal number of Diabetic Retina and Normal Retina. The selected images for this research work are in the form of .jpg format and all the images of *OCT* scan are in normal condition having no blurriness or fog. As the blurred images are not recognized by the physician as well as *DRuNN* based system. So, in this proposed work non-blurry images are considered for training and testing purpose. Once the dataset of images are gathered, it is converted into form of 1-D array or scaling the Pixel (as shown in Fig. 8) [21]. Afterwards with variation in dataset like 50%, 60%, 70%, 80%, and 90% images are reserved for training phase and rest images are reserved for testing phase.

The dataset contained images from patients of varying ethnicity, age groups and extremely varied levels of lighting in the retina. This affects the pixel intensity values within the images and creates unnecessary variation unrelated to classification levels. To counteract this, color normalization was implemented on the images. The images were also high resolution and therefore of significant memory size. The dataset was resized to 256x256 pixels which retained the intricate features.

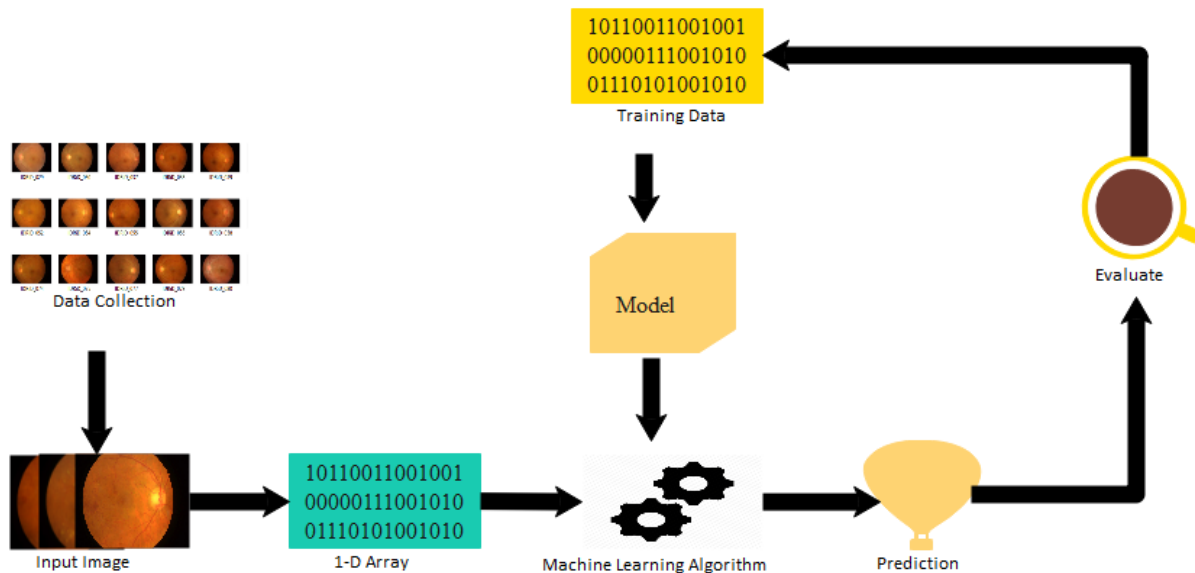


Fig. 8: Overview of Data Collection and Scaling of Pixels into 1-D array.

3.4 Feature Selection and Extraction

The feature selection is the second phase, where the features of various kind of images are used with the help of different algorithm like *CNN* and *ReLU*. The formal algorithm is used to the features of images [22]. The later algorithm helps to convert the image into two color form [23].

The importance of *CNN* and *ReLU* algorithm are that they study the feature of image and convert the image into only one color image. Further, the *VGG-16* model is used for training and reducing error for the developed model. Further to optimize the feature extraction, *Adam* optimizer is used. It supports power of adaptive learning rate and provides best accuracy prediction. The working of *CNN*, *ReLU* algorithms, *VGG-16* model, and *Adam* optimizer are described.

I. Convolutional Neural Network : To identify *OCT* based scan image, *CNN* algorithm plays an efficient role. It helps to study the feature of Image [22]. It generates the feature map with the help of its hidden layer (as shown in Fig. 9). The pseudo-code of *CNN* is shown in Algorithm 1.

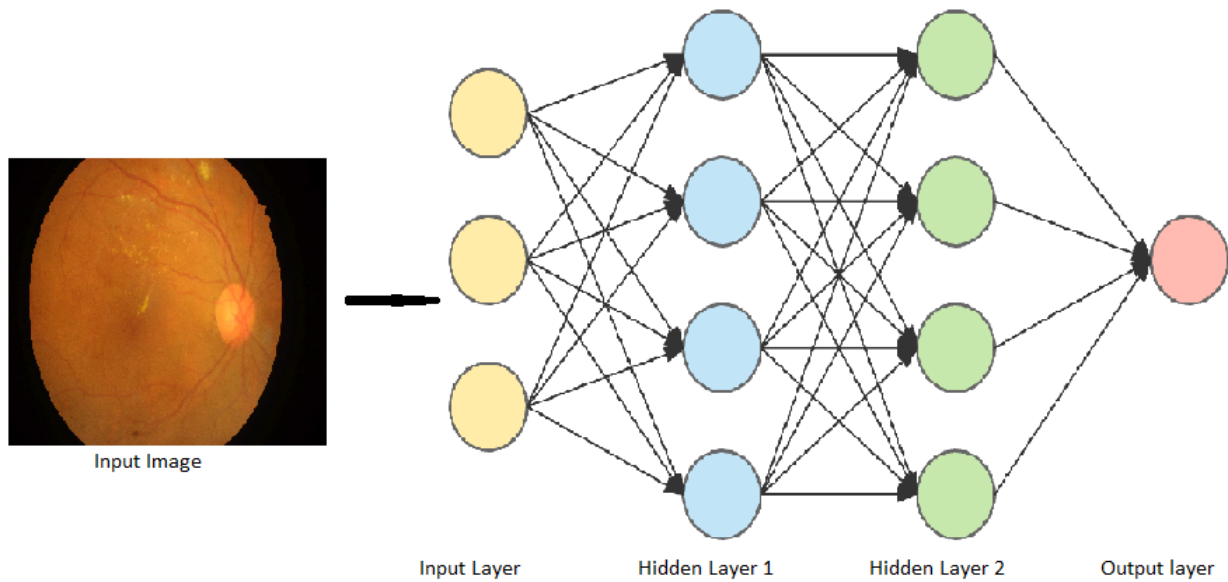


Fig. 9: Convolutional Neural Network Model

Algorithm 1 CNN Algorithm

Initialize the input image \mathbf{a}_i $\{i=1,2,3,\dots,n\}$

considering ac be the accumulator, and e be the

element value.

1: **for each** \mathbf{a}_x **in** \mathbf{a}_n :

2: **for each** p_i **in** p_x :

3: **set** $ac=0$

4: **for each** γ_x **in** γ :

5: **for each** e **in** the γ_x :

```

6:  if  $e \rightarrow p_i$ :
7:      return  $e * p_i$  and add result to ac
8:      end if
9:  end for
10: end for
11: end for
12: end for
13: set  $o_i = ac$ 

```

This algorithm take all horizontal rows of input image, and try to extracts the feature by pixel wise with a kernel filter. Afterwards, each element value is convolute with pixel value with addition of temporary accumulator which is initially zero. After convolutional, it generates the feature map (as shown in Fig. 10). The output image converted into 1-D array and helps to extracts the feature which is described into various subsection of *CNN*.

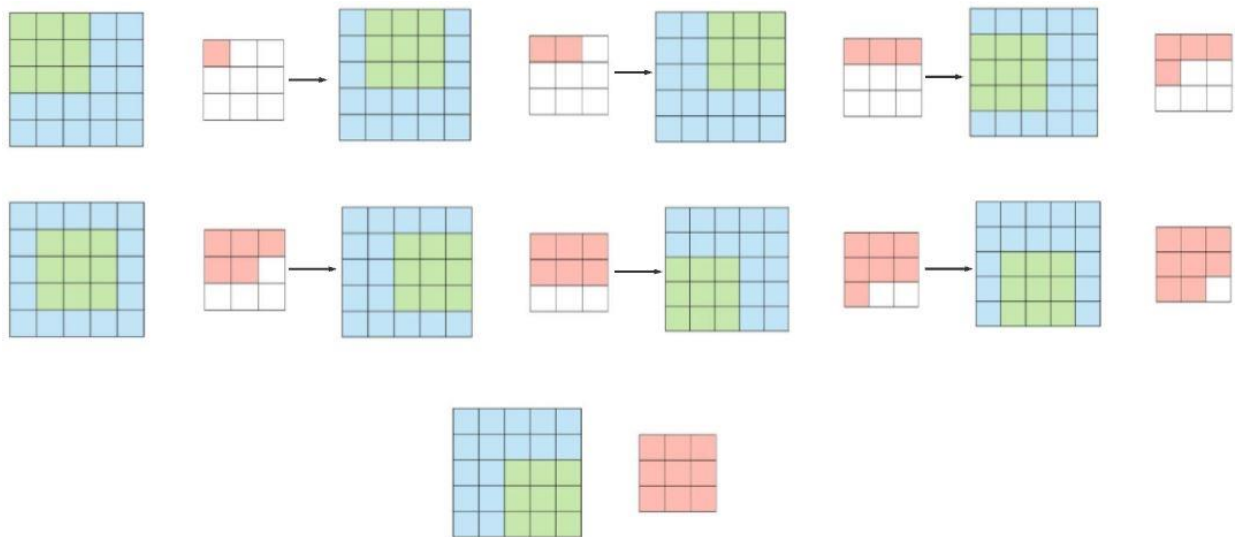


Fig. 10: Operations for generating Feature map

Layers on Convolutional Neural Network: The layers of *CNN* is helpful for generating the feature map firstly, conv2d layer is helpful for producing the tensor of outputs, pooling layer reduces the dimensionality of input image matrix. Further, Dropout layer drops some neurons to prevent the *DRuNN* technique from overfitting. Flatten layer flatten the output and generates the feature map. Afterwards, dense layer maintains the strong network between neurons of each layer to the each neurons of next layer. The layers in *CNN* model are shown in Fig. 11 and described by each step.



Fig. 11: Layers in *CNN*

a. **Convolutional 2-D:** Convolutional 2-D (*Conv2D*) layer plays an emerging role to create convolutional kernel that is wind with layers which is helpful for producing a tensor of Outputs [23]. Essential library Tensorflow 2.0 is installed in machine which is based on *DRuNN* technique. It is helpful for transferring the data in the form of array which is known as tensors (as shown in Fig. 12). Pseudo-code for algorithm 2 is shown below-

Algorithm 2 Convolutional -2D layer

Consider \vec{a}_x and $\vec{\gamma}_x$ be the input image vector and

kernel filter vector.

-
- 1: **for each** a_x **in** a_n :

 - 2: **for each** γ_x **in** γ :

 - 3: **set** $a_n = \vec{a}_x \cdot \vec{\gamma}_x$ // dot product

 - 4: **repeat** 1 to 3

 - 5: **end for**

 - 6: **end for**

 - 7: **return** a_n
-

This algorithm firstly performs the dot product between receptive field. i.e., input image vector and kernel vector. The result of the dot product is single integer of the output volume. Afterwards, the filter will slide over the next receptive field and generate a matrix, namely, tensor of outputs.

1	2	1	5	1	5
6	7	0	3	3	3
2	2	0	0	0	4
6	4	5	4	1	2
3	2	1	3	3	2
0	2	1	2	4	4

Fig. 12: Input Image in Tensor Form

b. **Pooling:**The extra structures slice of *CNN* algorithm is Pooling layer. This layer is beneficial for decreasing or resizing the dimensionality of an image without any loss of feature or pattern [24]. The computation as average pooling is used that takes the average value of each sample (as shown in Fig. 13). Algorithm 3 discussed the pseudo-code for pooling layer.

Algorithm 3 Pooling layer

Initialize the input image $a_i \{i=1,2,3,\dots,n\}$

considering w be the pooling window size
and s be

the stride. Set $s=1$ (initially)

1: **for each** a_x **in** a_n :

2: **for each** γ_x **in** γ :

3: **set** $w = (a_x / \gamma_x)$ // window size

4: **perform** $\left\lfloor \frac{a_n - w}{s} \right\rfloor + 1$

5: **end for**

6: **end for**

This algorithm firstly calculates the window size and after calculating window size it will compute the average of each grid. Average of each grids generates the vector in dimensionality reduce form.

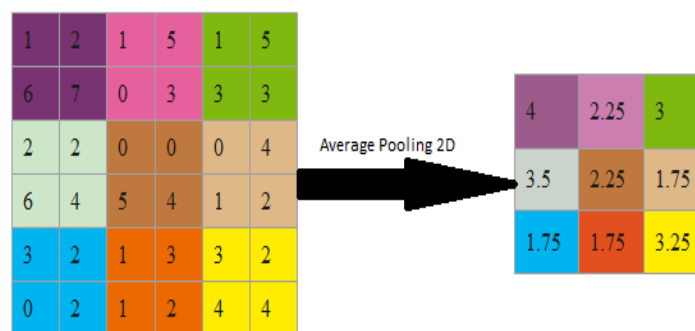


Fig. 13: Average Pooling

c. **Dropout:** The dropout layer automatically set the input units to zero with some frequency and some training rate. The main purpose of this layer is to prevent the *DRuNN* based technique from overfitting [25]. For the purpose some of the neurons are dropped.

In the proposed work, the images of *OCT* scan consists of black color at the boundary level which is not used for prediction. The black color can cause the model for over-fitting so dropout layer is used. Algorithm 4 discussed the pseudo-code of dropout layer.

Algorithm 4 Dropout layer

Suppose a neural network has H hidden layers $d \in$

$\{1, \dots, H\}$. \mathbf{a}^d be the vector of input, \mathbf{O}^d be the

vector of outputs. \mathbf{m}^d , \mathbf{B}^d be the weights and biases,

respectively, at layer, d ; and $f(a)$ be the activation

function

1: **for each** H :

2: **for each** m :

3: **for each** O :

4: **set** $a^{(d+1)} = m^{(d+1)}O^H + B^{(d+1)}$

5: **set** $O^{(d+1)} = f(a^{(d+1)})$

6: **end for**

7: **end for**

8: **end for**

This algorithm firstly drops some neuron with some pre-trained random learning. Afterwards, the generated activation function corresponds to the output layer which is helpful for predicting the category of scan.

d. **Flatten:** Flatten layer transforms the pooled trait plot to a lone column that is delegated to the fully connected sheet [26]. Flatten layer is helpful for converting *OCT* scan based eye image into the forms channel dimension. This layer flattens the output of convolutional layer to achieve a single and long characteristic vector which is known as feature map (as shown in Fig. 14). The feature map is used for pattern recognition for an Image. Algorithm 5 discussed the pseudo-code for flatten layer.

Algorithm 5 Flatten layer

Suppose a_x, a_y be the dimension of the input image,

and γ be the kernel filter.

```
1:  for each  $a_x, a_y$  in  $a_n$ :
2:    set  $(a * \gamma)[a_x, a_y]$ 
3:      for  $i$  in 1 to  $n$ :
4:        for  $j$  in 1 to  $n$ :
5:          set  $\sum_i \sum_j \gamma[i, j] a[a_x - i, a_y - j]$ 
6:        end for
7:      end for
8:    end for
```

This algorithm helps to traverse the input image matrix in row as well as column form with kernel filter. Afterwards, the convolutional operation on input matrix and kernel filter, it generates another matrix, which is called feature map.

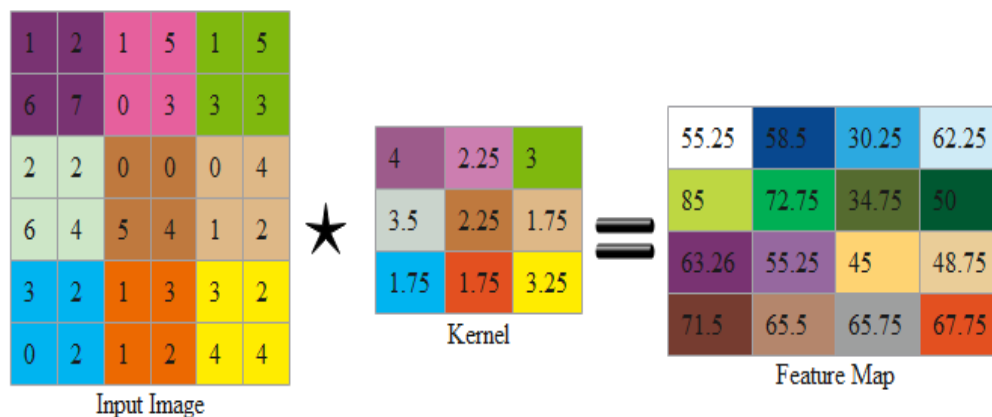


Fig. 14: Feature Map

e. **Dense:** This layer is helpful for creating link between the convolutional network and the target class [27]. In dense layer each neuron of previous state is fully connected with the neuron of next state. The dense layer is the set of nodes that creates connection between the layers. The keras library automatically handles the connection between the layers. This layer is used after the pooling layer for mapping of features in a row or a column. Algorithm 6 discussed the pseudo-code for dense layer.

Algorithm 6 Dense layer

Suppose m be the weight, B be bias, x , and y be the

row and column of feature matrix, and O be the

output. Initialize $tmp=0$

1: **for each** i in 1 to x :

2: **set** $tmp=0$

3: **for each** j in 1 to y :

4: $tmp = tmp + m[i][j] \times B[j]$

5: $tmp = tmp \times B[j]$

6: **end for**

7: $O[i] = tmp$

8: **end for**

This algorithm maintains the strong network between neurons of each layer to the neurons of next layer and updates the weight matrix corresponding to each neuron.

II. **Rectified Linear Unit:** *ReLU* is a protocol of non-linear or activation layer. *ReLU* model is used immediately after each *CNN* layer [28]. The leading aim of *ReLU* model is to initiate non linearity into neural network model which have been computed as linear. *CNN* model converts the image into black and gray color. Afterwards, *ReLU* model works only in a single color either black or gray. For gray color, it gives the positive value and zero for black color (as shown in Fig. 15). This approach reduced the feature of image.

The mathematical approach is shown as –

$$f(a) = \begin{cases} 0 & a < 0 \\ a & a \geq 0 \end{cases} \quad (13)$$

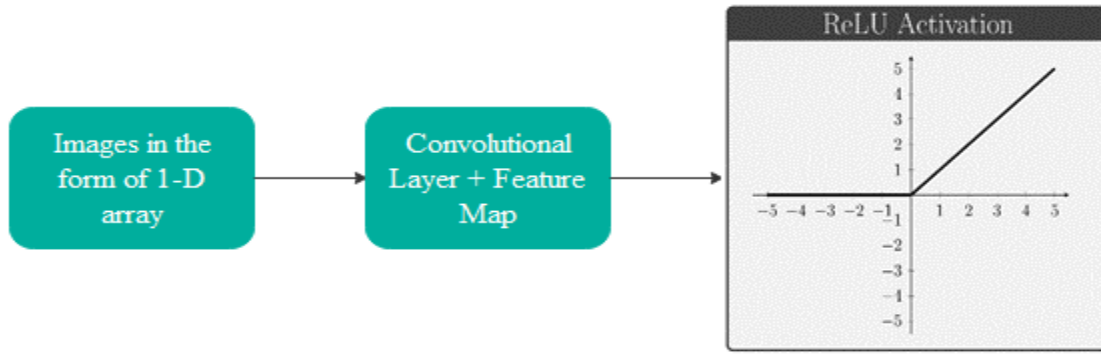


Fig. 15: *ReLU* Model Architecture

III. Visual Geometry Group-16 Model: Visual Geometry Group-16 (vgg-16) based on Imagenet architecture and very similar to Alexnet architecture [29][30][31]. Vgg-16 model increase the number of features which expand the depth of the network. Vgg-16 model consists of 3 x 3 meaningful small kernel size and the vgg-16 network comprise the 138 million parameters (as shown in Table 2). For small number of images vgg-16 model plays an efficient role. In this proposed research work 3 channel based RGB image of size 224 x 224 is passes through the DRuNN based technique (as shown in Fig. 16). The succession of 3 x 3 convolutional layer still introduced the non-linearity which leads to the greater discrimination power to the network.

Let a be the input feature map of image and F be the output feature map, n , and r be the width and height of filters then mathematical approach for calculating number of parameters can be given as-

$$parameters \# = (n * r * a + 1) * F \quad (14)$$

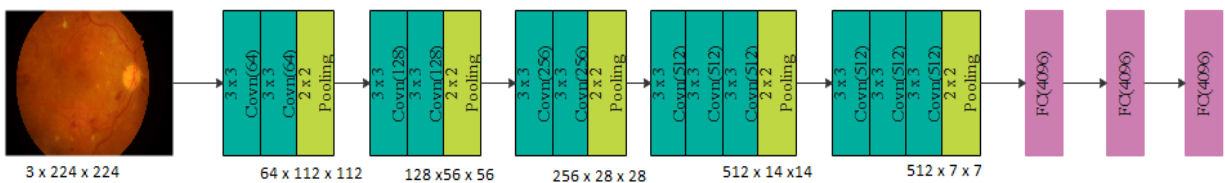


Fig. 16: Working of Visual Geometry Group-16 Model

Table 2
VGG-16 Model with training Parameters

Padding and Stride	Input Dimensions	Activation Shape	Parameters #
	(3 x 224 x 224)		
$z=1, s=1$	Conv3-64	224*224*64	$(3*3*3+1)64 +$
	Conv3-64	224*224*64	$(3*3*64+1)64$

s=2	Pooling-1	112*112*64	0
z=1, s=1	Conv3-128	112*112*128	$(3*3*64+1)*128 +$
	Conv3-128	112*112*128	$(3*3*128+1)*128$
s=2	Pooling-2	56*56*128	0
z=1, s=1	Conv-256	56*56*256	$(3*3*128+1)*256 +$
	Conv-256	56*56*256	$(3*3*256+1)*256 +$
	Conv-256	56*56*256	$(3*3*256+1)*256$
s=2	Pooling-3	28*28*256	0
z=1, s=1	Conv3-512	28*28*512	$(3*3*256+1)*512 +$
	Conv3-512	28*28*512	$(3*3*512+1)*512 +$
	Conv3-512	28*28*512	$(3*3*512+1)*512$
s=2	Pooling-4	14*14*512	0
z=1, s=1	Conv3-512	14*14*512	$(3*3*512+1)*512 +$
	Conv3-512	14*14*512	$(3*3*512+1)*512 +$
	Conv3-512	14*14*512	$(3*3*512+1)*512$
s=2	Pooling-5	7*7*512	0
	FC-4096	(4096,1)	$4096*(7*7*512) + 4096$
	FC-4096	(4096,1)	$4096*(4096) + 4096$
	FC-4096	(4096,1)	$1000*(4096) + 1000$
Total			approximately 138M

The Table 2 uses the five successive convolutional layer 3 fully connected layer. The input image is of *RGB* channel having 224 x 224 dimensions. As *vgg-16* model comprises of 138M parameters. So, the Table 2 calculates the number of parameters at each layer with the help of equation (14). After calculating the number of parameters it shows that the total result is 138 millions.

IV. Adaptive Moment Estimation Optimizer: In this proposed research work, Adaptive Moment estimation (*Adam*) optimizer is used for minimization purpose of errors [32]. Adam optimizer is a combination of momentum and Root Mean Square (*RMS*) prop [33]. This

optimizer supports the power of adaptive learning rate. This algorithm can also run on the noisy environment and gives the best accuracy. Adam optimizer reduces the error and update the model (as shown in Fig. 17).

The pseudo-code of Adam optimizer is shown in Algorithm 7.

Algorithm 7 *Adam optimizer reducing Error*

α be the stepsize and $\beta_1, \beta_2 \in [0,1)$ be the exponential rates for moment estimates. $f(\theta)$ be the stochastic objective function with parameters θ . θ_0 be the initial parameter vector.

$\Phi_0 \leftarrow 0$ (Initialize 1st moment vector)

$\mathbf{v}_0 \leftarrow 0$ (Initialize 2nd moment vector)

$t \leftarrow 0$ (Initialize timestep)

-
- 1: **While** θ_t not converged do

 - 2: $t \leftarrow t + 1$

 - 3: $\mathbf{g}_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ // get gradients

 - 4: $\Phi_t \leftarrow \beta_1 \cdot \Phi_{t-1} + (1 - \beta_1) \cdot \mathbf{g}_t$ //update biased first moment estimation

 - 5: $\mathbf{v}_t \leftarrow \beta_2 \cdot \mathbf{v}_{t-1} + (1 - \beta_2) \cdot \mathbf{g}_t$
//update biased second moment estimation

 - 6: $\hat{\Phi}_t \leftarrow \Phi_t / (1 - \beta_1^t)$ // compute bias corrected first moment estimation.

 - 7: $\hat{\mathbf{v}}_t \leftarrow \mathbf{v}_t / (1 - \beta_2^t)$ // compute bias corrected second moment estimation.

 - 8: $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{\Phi}_t / ((\text{sqrt}(\hat{\mathbf{v}}_t) + \epsilon))$
//updating parameters

 - 9: **end While**

 - 10: **return** θ_t //Resulting parameters

This algorithm helps to reduce the error rate on proposed *DRuNN* system. Firstly, gradient descent maintains a single learning rate for all weight updates and the learning rate does not alter during training. As *Adam* optimizer is a combination of *RMS* prop, AdaGrad algorithm so *Adam* also maintain the property of these algorithm. As *RMS* prop maintains the pre-parameter learning rates that are adapted based on average of first moment vectors. Similarly, *Adam* optimizer also maintains the average of first as well as second moment vectors which makes it efficient for calculating error in noisy environment.

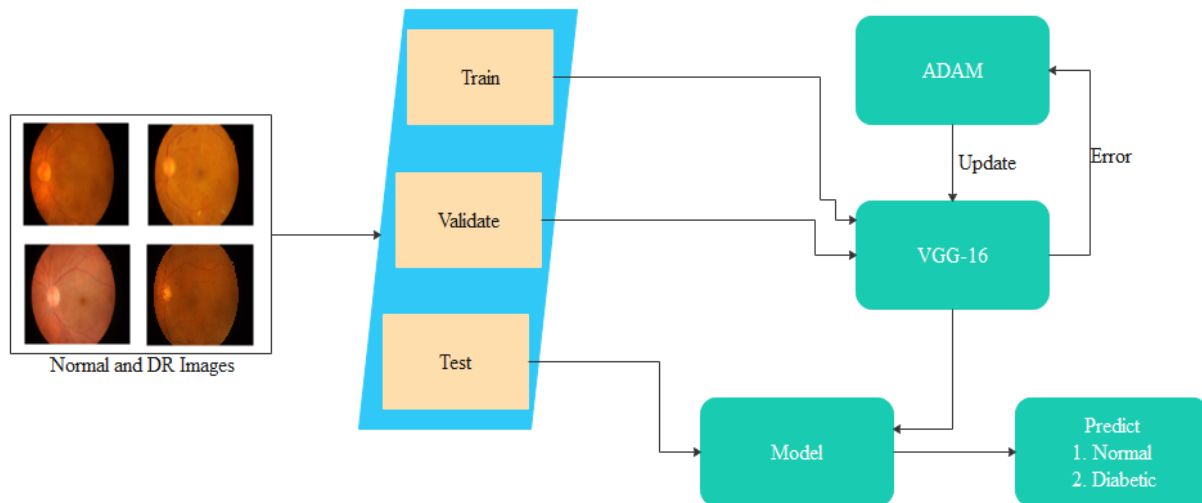


Fig. 17: Working of *Adam* optimizer Reducing Error

3.5 Classification

This is an important and final step, where prediction is done. For the purpose, a GUI based platform is developed for the user's suitability [34]. When the GUI file is compiled then a *GUI* based new window is opened. The window consists of two button "Select" and "detect". After clicking the "Select" button, the folder is opened where dataset of images are stored. Once, the image is selected then after clicking on the "Detect" button, the machine will predict the selected image as of *DR* affected patients or either normal patient. If the image is belongs to the normal patient category then the message Normal will be appeared on the image and otherwise Diabetic will be appeared

CHAPTER 4

(PERFORMANCE ANALYSIS)

4.1 System Requirement:

The experiments in this research are performed on 64-bit windows operating system consisting Intel(R) Core (TM) i5-7200U CPU @ 250GHZ Processor. The installed memory size is 4 GB.

4.2 Library Requirement:

4.2.1 NumPy:

NumPy stands for Numeric Python. It is used for scaling the pixel and converting the *OCT* scan in 1-D array and used for basic mathematical array arithmetic operations.

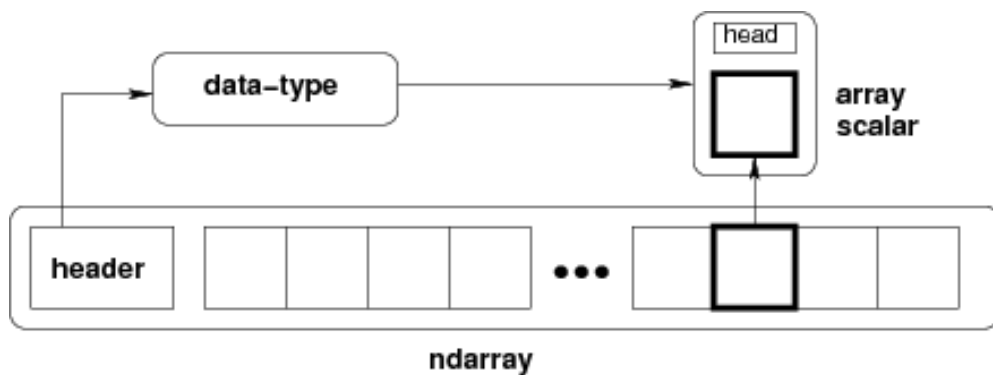


Fig. 18 NumPy Library working

4.2.2 Pandas:

Pandas is one of the most significant python based programming library, that is used to import the path means dataset of mages in any machine learning based project. It can be expressed as

```
import Pandas as pd # typecasting the pandas name as pd
```

```
X = pd.read_csv("path_name/Location")
```

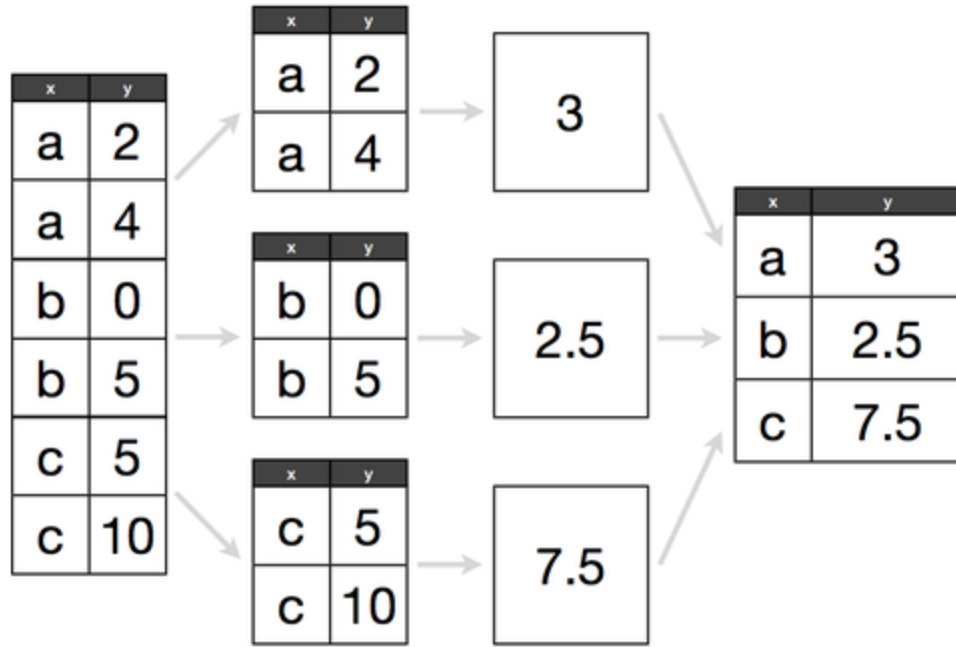
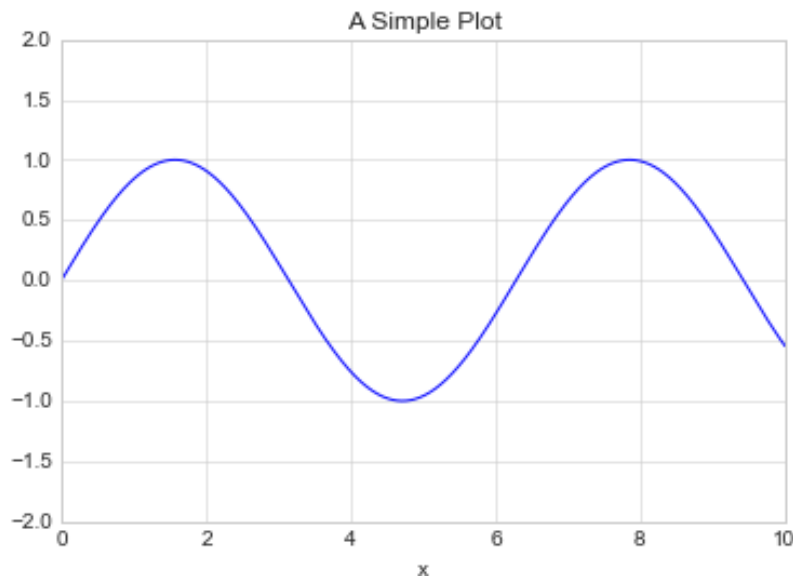


Fig. 19 Pandas Library Internal Working

4.2.3 Matplotlib:

It is a basic library for data visualization. This library is used when we have to plot our data means when we have to see the graphical representation of data. The matplotlib library provides some in built graph such as histogram, bar graph, pie chart, and line chart.



Graph. 1 Simple plot by Matplot Library

4.2.4 skLearn:

skLearn stands for scikit-learn. It is a open source library that has powerful tools for data analysis and data mining. It supports Python mathematical and factual libraries like NumPy and SciPy.

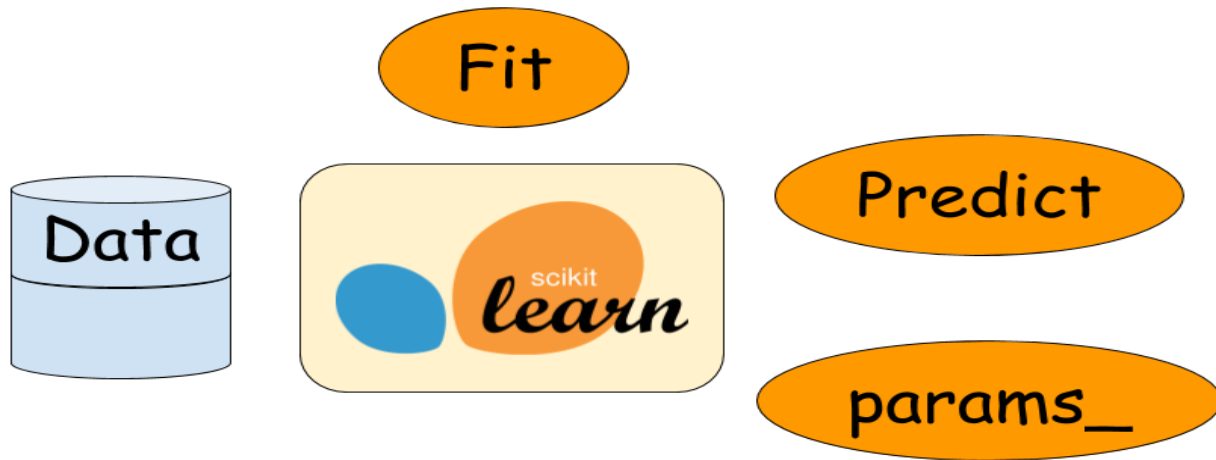
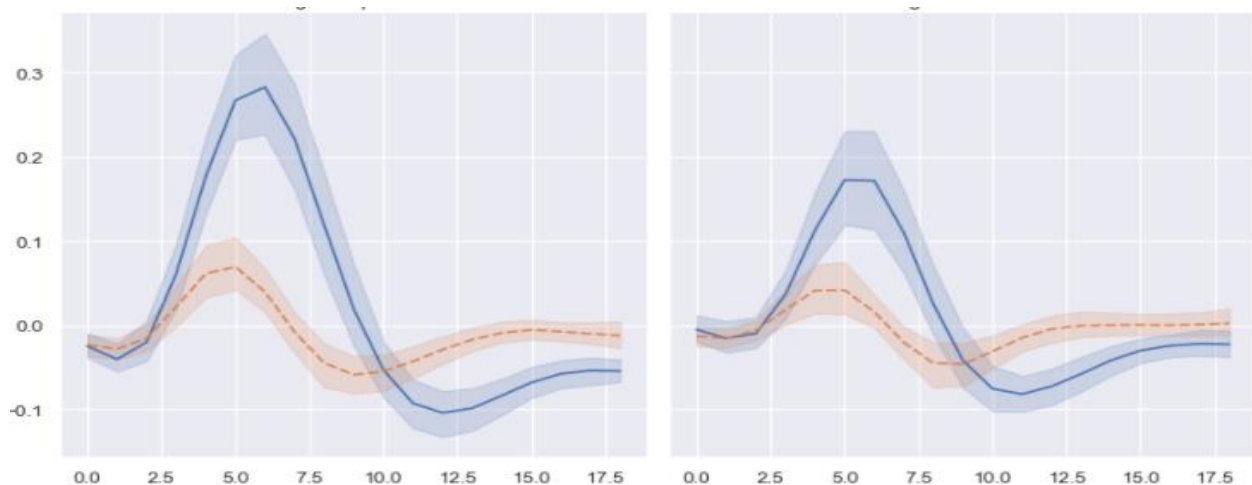


Fig. 20 Flow of skLearn Library

4.2.5 Seaborn:

It is a Python data visualization library based on matplotlib. It is just extension of matplot lib when we have to plot an attractive graph.



Graph. 2 Attractive Graph by Seaborn Library

4.2.6 Tensorflow:

Tensorflow is one of the most important library for the proposed work which is provided by Google brain team and main aim of tensorflow library is to compute large scale mathematical computations. The dataset which is converted into the tensor of output is handled by tensorflow library. Keep remember that the proposed *DRuNN* technique used tensorflow 2.0 library which can be installed on python 3.7.x version.

3.2	1.4	5.1	-
-1	-2	2.4	-
-	-	-	-
-	-	-	-

Tensor

Fig. 21 Tensor in Array Format

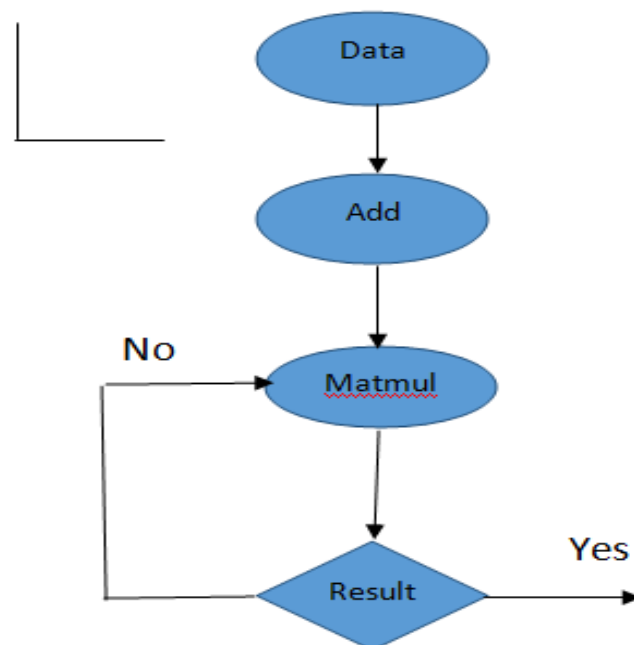


Fig. 22 Flow Diagram of Tensorflow Library

4.2.7 Keras:

Keras is a high source library which is used just wrapping the tensorflow library. In this research based project keras library is just used before the tensorflow library *i.e.* keras.tensorflow.preprocessing.xxx.

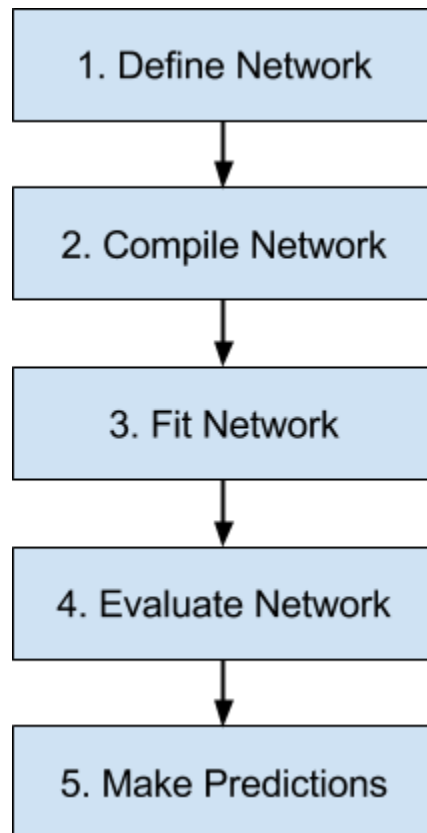


Fig. 23 Life Cycle of Neural Network in Keras

4.2.8 OpenCv :

Open Source Computer Vision Library which used to operate the computer vision related task. This library is developed by intel and later on with the help of google team this library stands a good role in vision related project. The main aim of this library is to recognize the images.

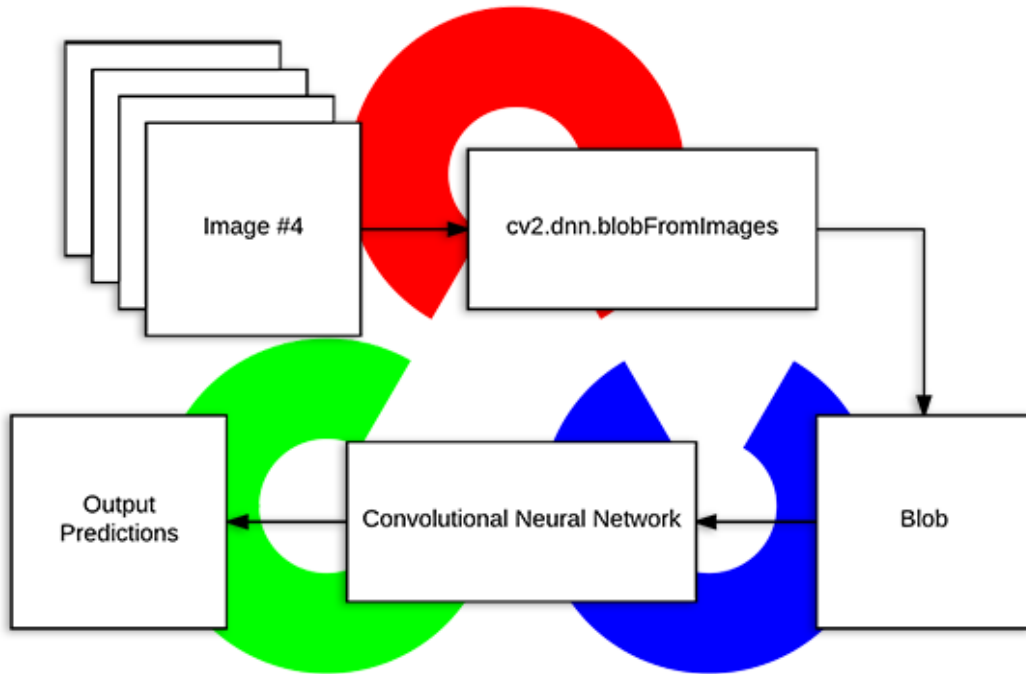


Fig. 24 Working of OpenCv Library

4.2.9 tkinter Library:

Tkinter library is used for developing a GUI based system.

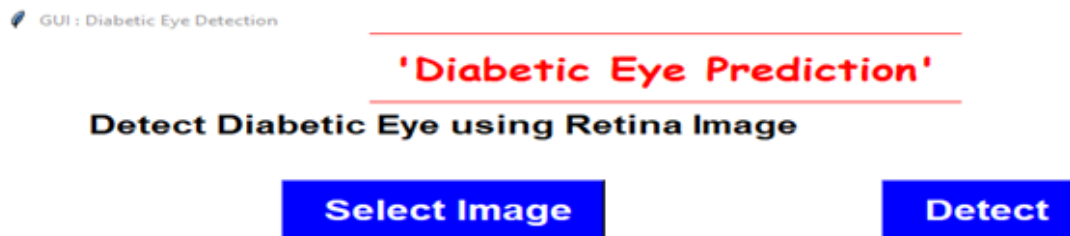


Fig. 25 GUI based Window

4.3 System Development Approach

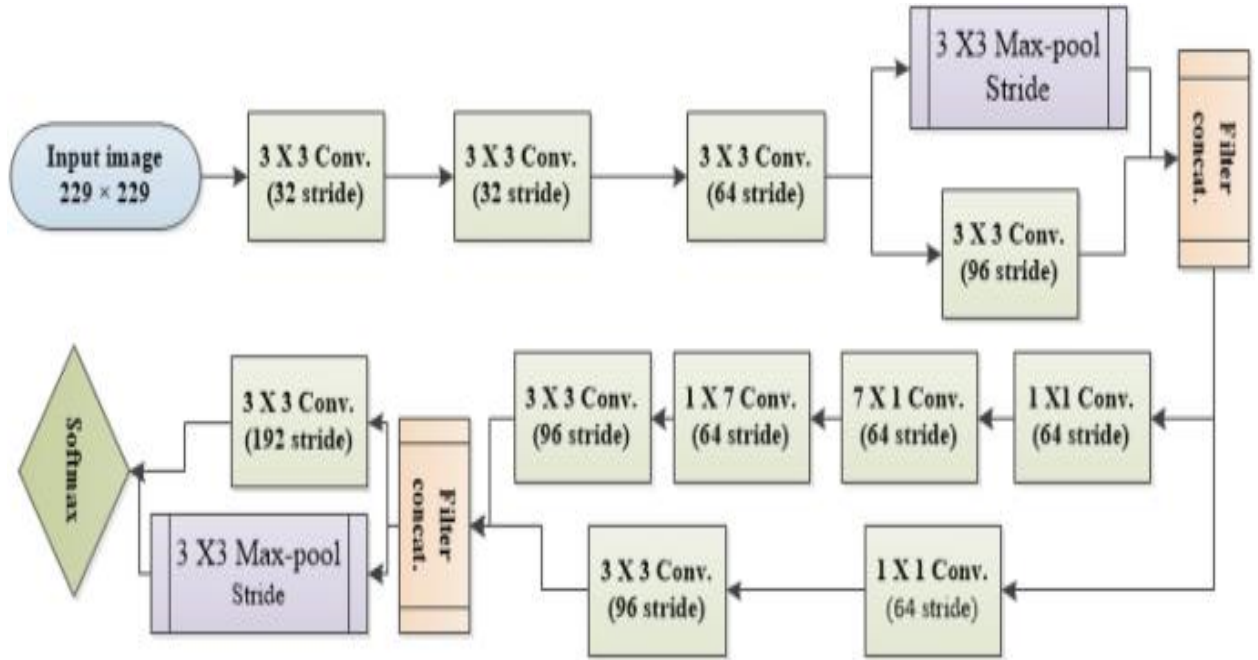


Fig. 26 Image Classification

4.3.1 Sequence Diagram:

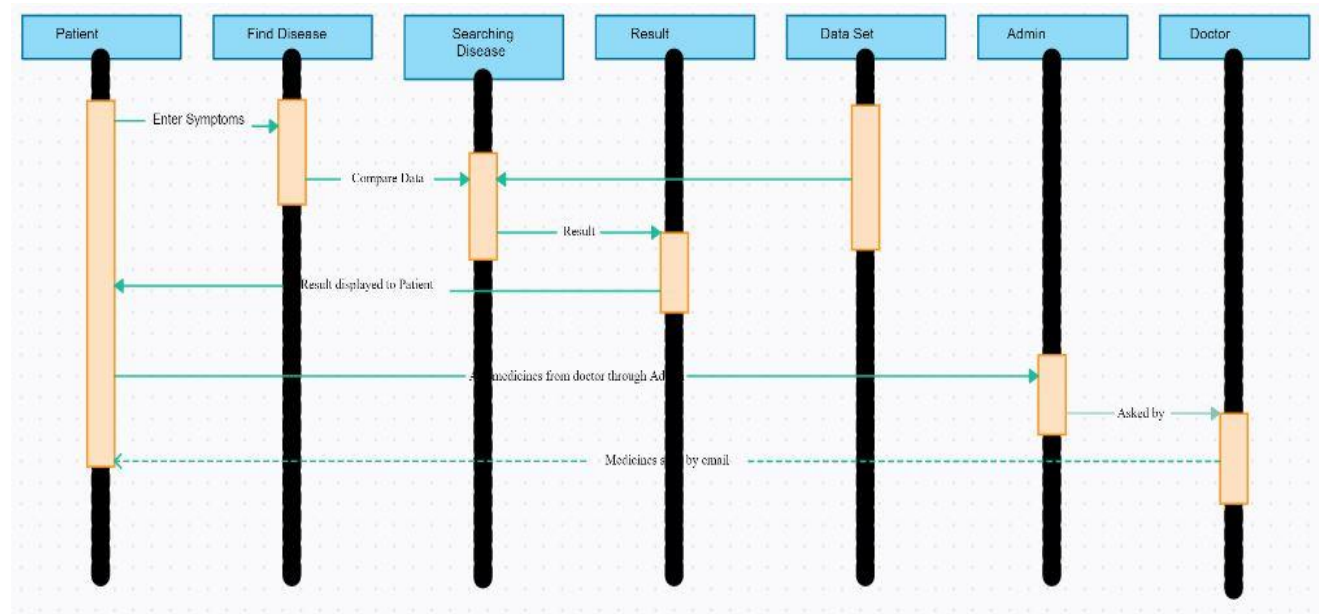


Fig. 27 Sequence Diagram

4.3.2 Use Case Diagram:

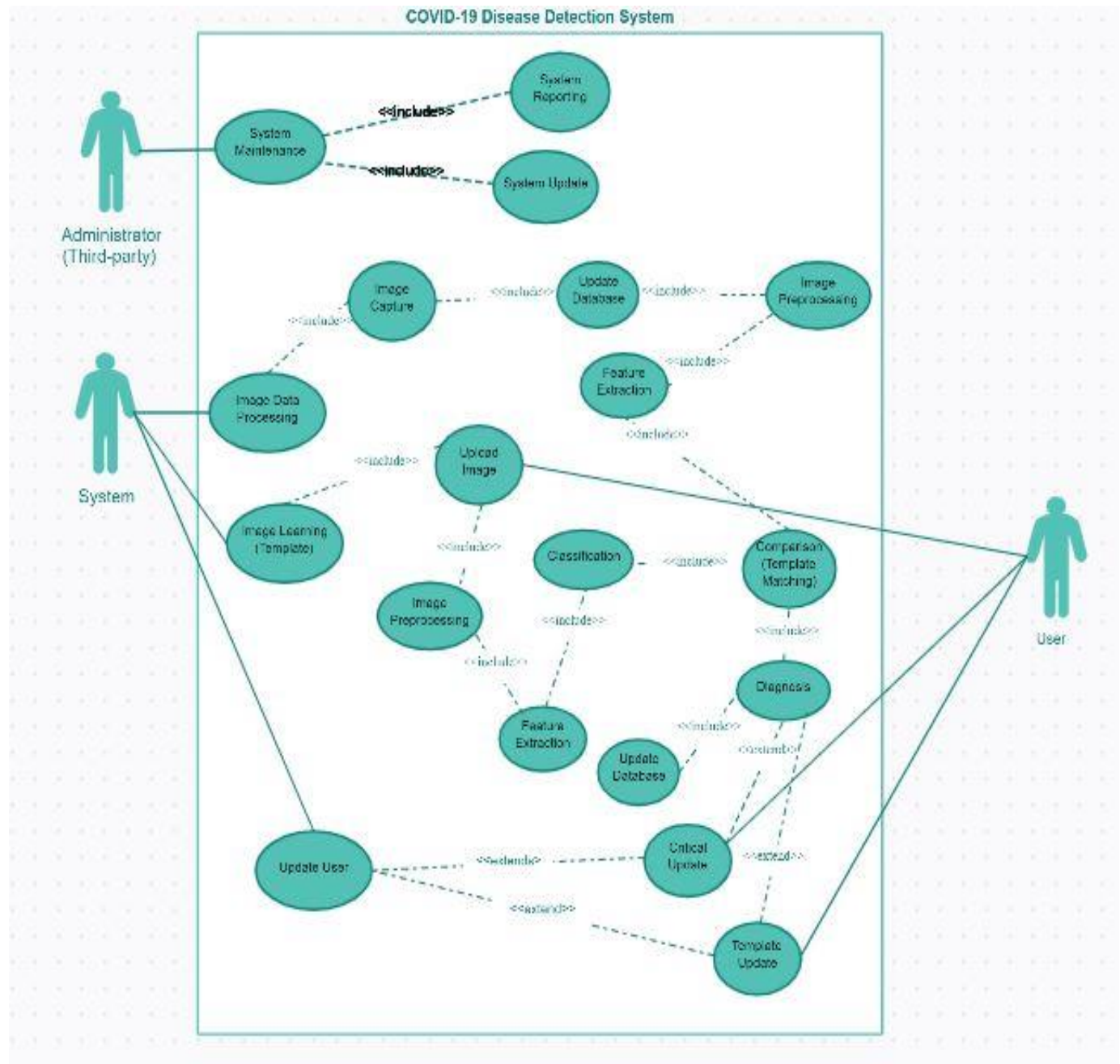


Fig. 28 Use Case Diagram

4.3.3 Class Diagram:

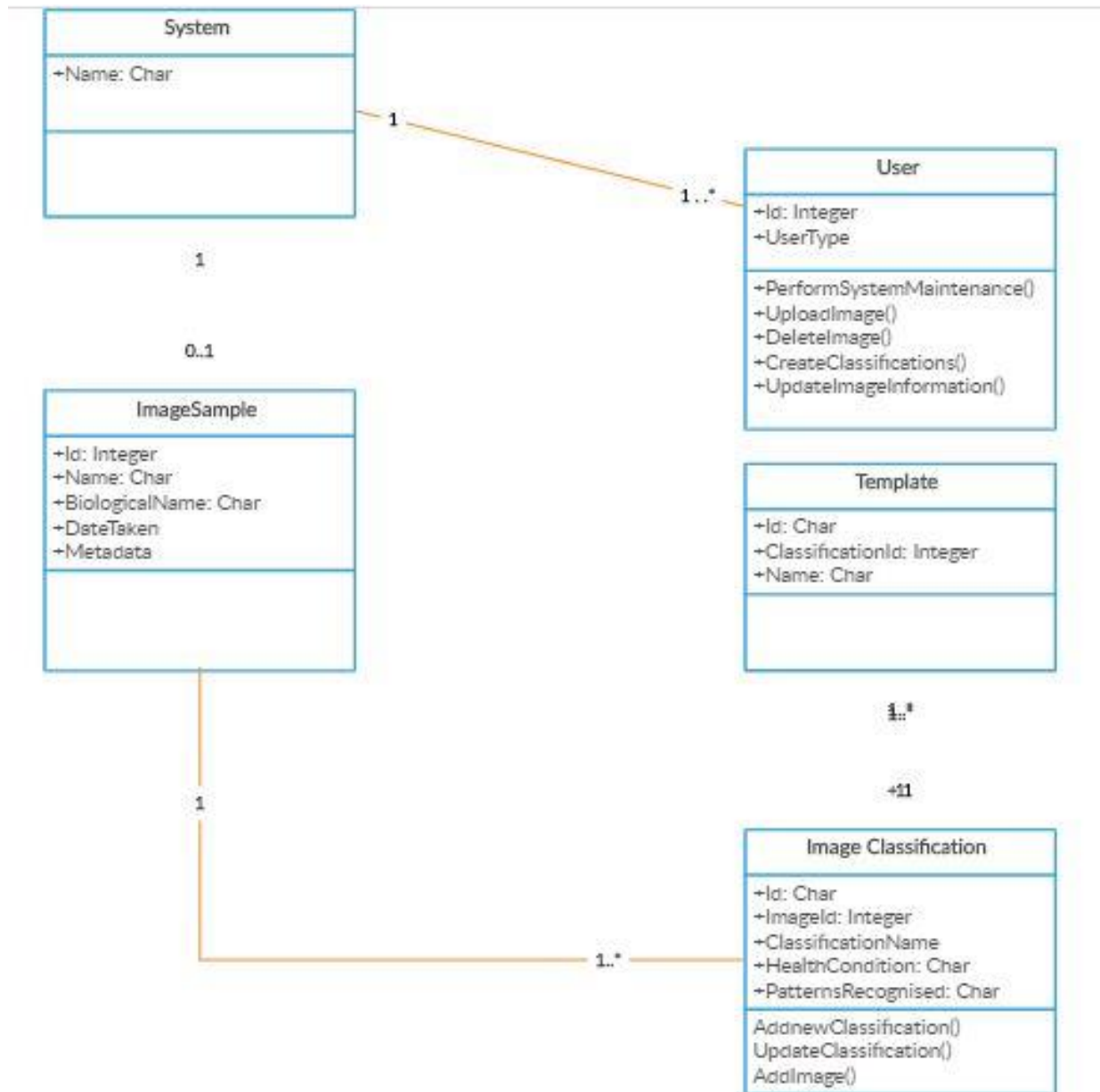


Fig. 29 Class Diagram

4.3.4 Activity Diagram:

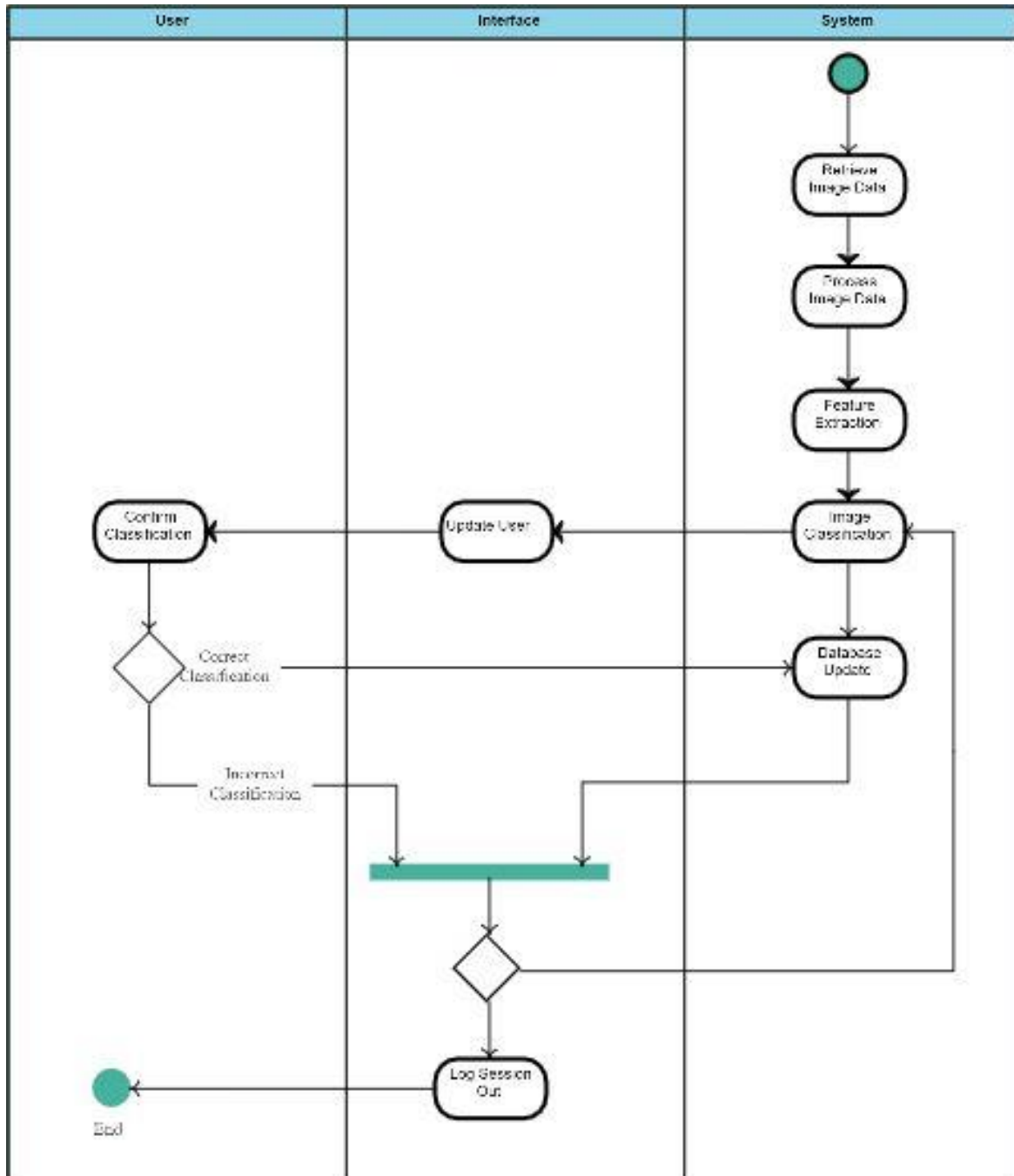


Fig. 30 Activity Diagram of DRuNN Technique

4.3.5 Data Flow Diagram:

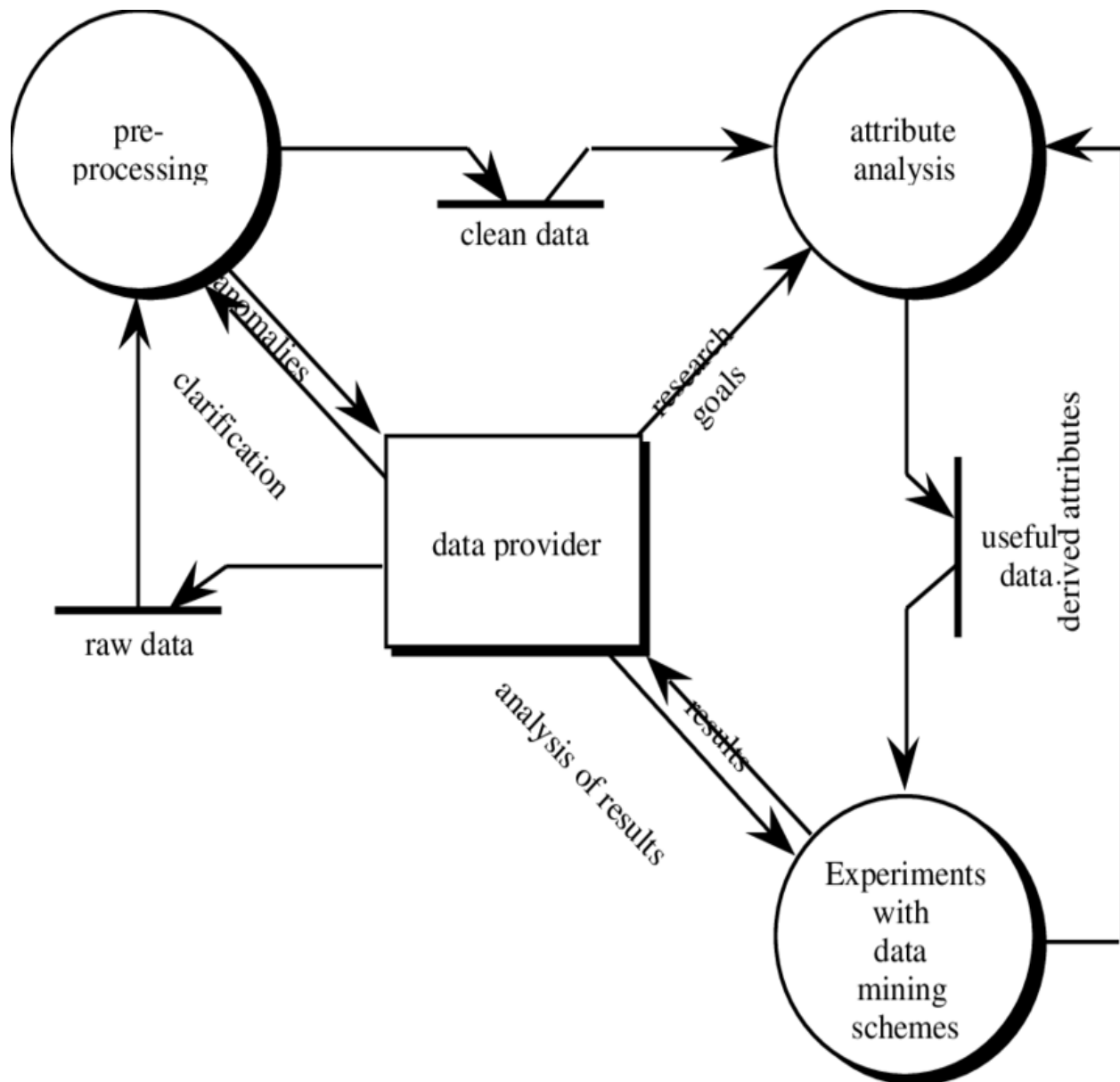
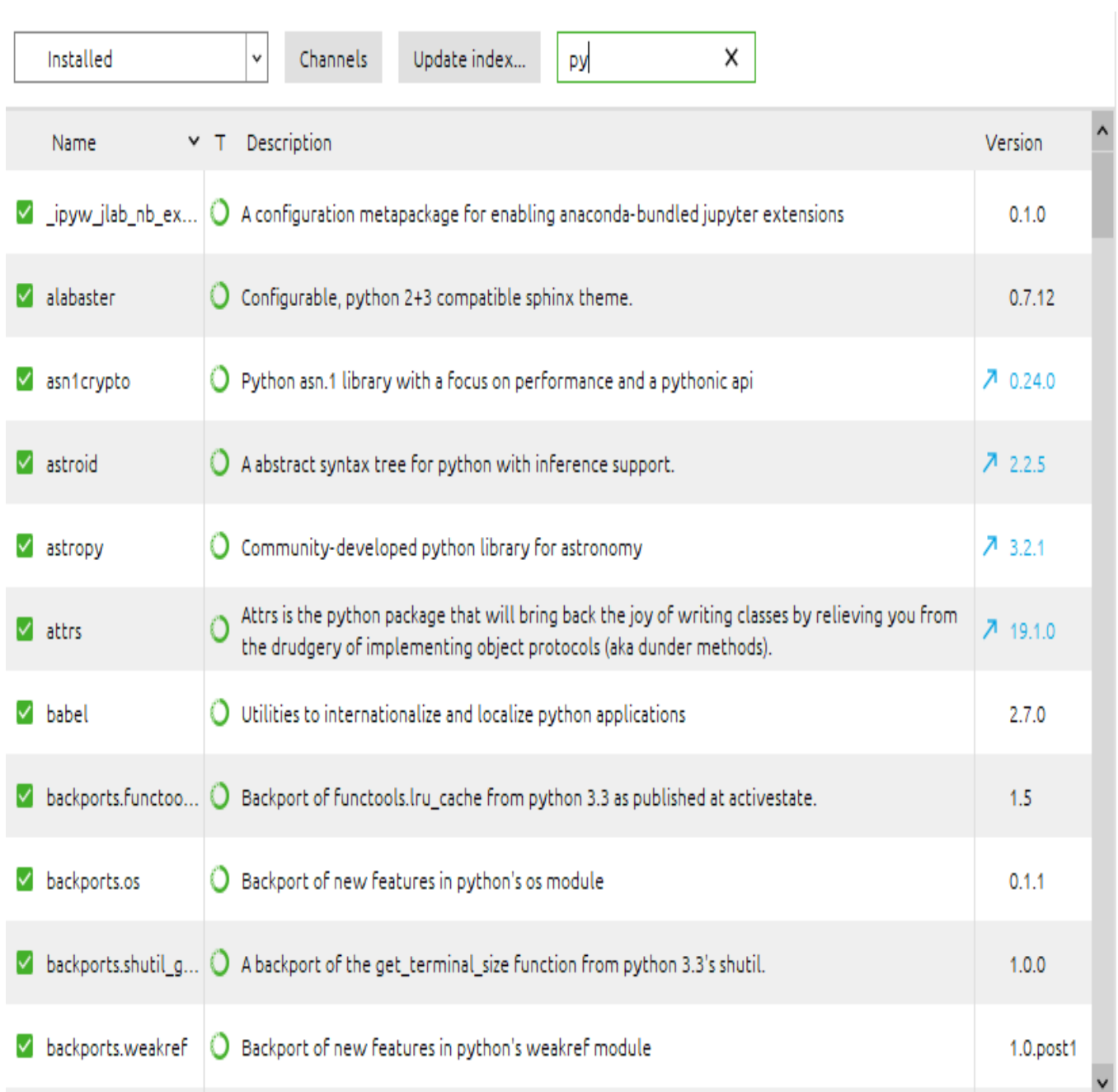


Fig. 31 Data Flow Diagram

4.4 Installation of Packages



Installed Channels Update index... py X

Name	T	Description	Version
<input checked="" type="checkbox"/> _ipyw_jlab_nb_ex...		A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
<input checked="" type="checkbox"/> alabaster		Configurable, python 2+3 compatible sphinx theme.	0.7.12
<input checked="" type="checkbox"/> asn1crypto		Python asn.1 library with a focus on performance and a pythonic api	0.24.0
<input checked="" type="checkbox"/> astroid		A abstract syntax tree for python with inference support.	2.2.5
<input checked="" type="checkbox"/> astropy		Community-developed python library for astronomy	3.2.1
<input checked="" type="checkbox"/> attrs		Attrs is the python package that will bring back the joy of writing classes by relieving you from the drudgery of implementing object protocols (aka dunder methods).	19.1.0
<input checked="" type="checkbox"/> babel		Utilities to internationalize and localize python applications	2.7.0
<input checked="" type="checkbox"/> backports.functoo...		Backport of functools.lru_cache from python 3.3 as published at activestate.	1.5
<input checked="" type="checkbox"/> backports.os		Backport of new features in python's os module	0.1.1
<input checked="" type="checkbox"/> backports.shutil_g...		A backport of the get_terminal_size function from python 3.3's shutil.	1.0.0
<input checked="" type="checkbox"/> backports.weakref		Backport of new features in python's weakref module	1.0.post1

Fig. 32 Installation of Packages

4.5 Discussion of Results and Comparison

4.5.1 Experimental Setup

The experiments in this research are performed on 64-bit windows operating system consisting Intel(R) Core (TM) i5-7200U CPU @ 250GHZ Processor. The installed memory size is 4 GB.

4.5.2 Data Collection and Preprocessing

For developing *DRuNN* technique the model is trained with the variation of 50%, 60%, 70%, 80%, and 90% dataset values. Afterwards, scaling of pixels means the input image is converted into the form of 1-D array. Table 3 describes the simulation parameters of proposed framework.

Table 3

Simulation Parameters

S. No	Attributes	Values	Specification
1.	Algorithm	<i>CNN, ReLu</i>	<i>CNN</i> Model is used to study the curve, edge, and color filter. Relu model is helpful for introducing non linearity and converting <i>RGB</i> image into binary image.
2.	Library	Numpy, Tensorflow 2.0, Keras, tkinter	Numpy is used for converting an Image into 1-D array, Tensorflow is used for numerical computation of model, Keras is used for evaluating deep learning model, and tkinter library is used for developing <i>GUI</i> based screen.
3.	Layers	<i>Conv2d, Dropout, pooling, Flatten, Dense</i>	<i>Conv2d</i> is helpful for producing tensor of outputs, <i>Dropout</i> is used for preventing the model from overfitting, <i>pooling</i> layer is used for reducing dimensionality, <i>flatten</i> layer is used for making feature map, and <i>Dense</i> is used for creating link between convolutional network and target class.
4.	Model	VGG-16	<i>VGG-16</i> is 16 layers architecture which is used for the training purpose of <i>DRuNN</i> technique.
5.	Optimizer	Adam	Adam optimizer is used to reduce the error rate, this optimizer is also work on noisy environment.
6.	Accuracy	90.00%	Calculated as: $\frac{TruePositive + TrueNegative}{TotalNumberofSamples}$

7.	Error	10.00%	Calculated as: $\frac{FalsePositive + FalseNegative}{TotalNumberofSamples}$
8	Specificity	86.67%	Calculated as: $\frac{TrueNegative}{TrueNegative + FalsePositive}$
9.	Sensitivity	94.28%	Calculated as: $\frac{TruePositive}{TruePositive + FalseNegative}$

4.5.3 Architecture used in Feature Selection

For extracting feature of the OCT scans, *CNN* algorithm is applied and with the help of *CNN* the important features like color, curve, and edge of the image is studied. The *CNN* algorithm is comprises with input layer, hidden layers, and output layer on which forward as well as backward propagation is applied for maintaining the strong network between the layers.

The convolutional layers helps to create the feature map. Conv-2d layer creates the tensor of output matrix. The tensor of output matrix is reduced with the help of Average Pooling which takes the average of each grid. Afterwards, tensor matrix and pooling matrix performs the convolution to generate the feature matrix.

Futher, *Relu* model is helpful for introducing the non linearity as well as it converts the image into one color image. Afterwards, the various ImageNet model like Inception-V1, Inception-V2, AlexNet, Inception-V3, and *VGG-16* models are used for training perspective with the variation of dataset values.

The achieved highest accuracy are 48.24%, 57.31%, 68.82%, 81.02%, 90.00% respectively. Due to its simple network, *VGG-16* model shows the highest accuracy value which is adopted for proposed *DRuNN* system. (as shown Fig. 33). Further, for reducing error rate *Adam* optimizer is used which iteratively reduced the error rate and updates the model.

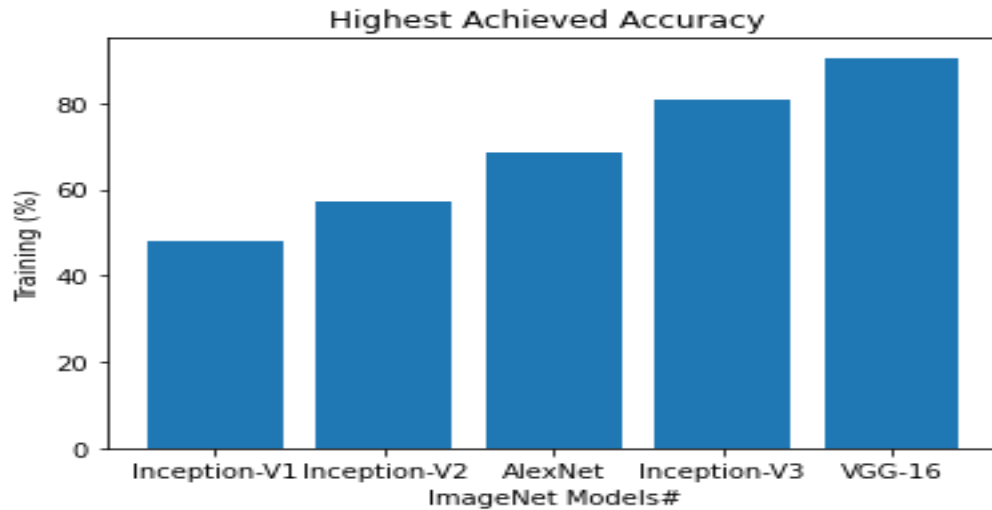


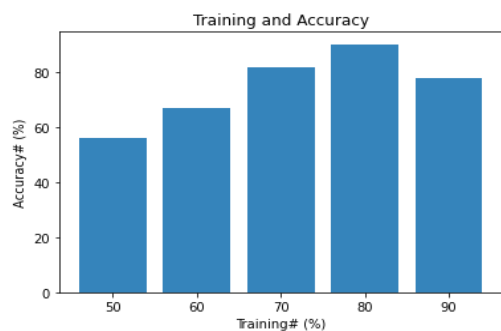
Fig. 33: Comparison of accuracy with other existing ImageNet models

As *VGG-16* model achieves the highest accuracy prediction. Therefore, the value of other performance parameters are calculated on *VGG-16* model. After predicting accuracy, error rate is calculated on developed *DRuNN* system. Afterwards specificity is calculated which tells about the negative case which is also got predicted as negative. Further, sensitivity of *DRuNN* system is calculated which tells about the positive case which is also got predicted as positive. Fig 34 compares the results value of performance parameters. Table 4 described the parameters of accepted transfer Learning Model.

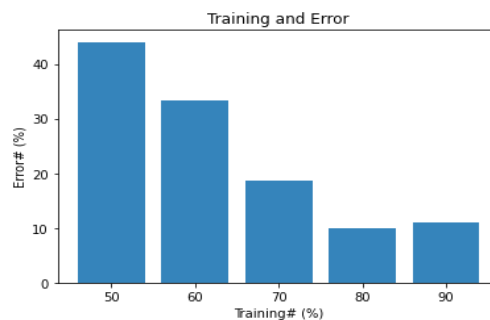
Table 4
VGG-16 model with variation in Training dataset values

Samples (100)	Confusion_Matrix	Accuracy (in %)	Error (in %)	Specificity (in %)	Sensitivity (in %)								
Training (50%) Samples (50)	<table border="1"> <tr> <td></td> <td colspan="2">Actual</td> </tr> <tr> <td rowspan="2">Predicted</td> <td>12</td> <td>13</td> </tr> <tr> <td>9</td> <td>16</td> </tr> </table>		Actual		Predicted	12	13	9	16	56.00	44.00	55.17	57.14
	Actual												
Predicted	12	13											
	9	16											
Training (60%)	<table border="1"> <tr> <td></td> <td colspan="2">Actual</td> </tr> <tr> <td rowspan="2">Predicted</td> <td>16</td> <td>12</td> </tr> <tr> <td>8</td> <td>24</td> </tr> </table>		Actual		Predicted	16	12	8	24	66.67	33.33	66.67	66.67
	Actual												
Predicted	16	12											
	8	24											

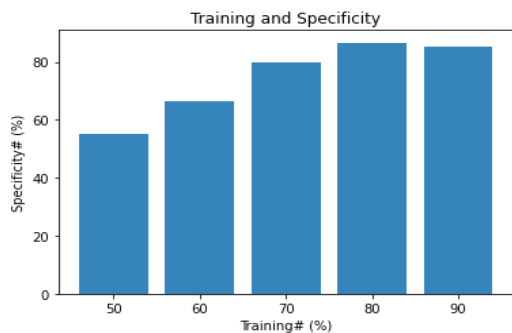
Samples						
(40)						
Training	Actual					
(70%)						
Samples	Predicted	25	8			
(30)		5	32	81.42	18.57	80.00
						83.33
<hr/>						
Training	Actual					
(80%)						
Samples	Predicted	33	6			
(20)		2	39	90.00	10.00	86.67
						94.28
<hr/>						
Training	Actual					
(90%)						
Samples	Predicted	30	7			
(10)		3	40	77.78	11.11	85.10
						90.90



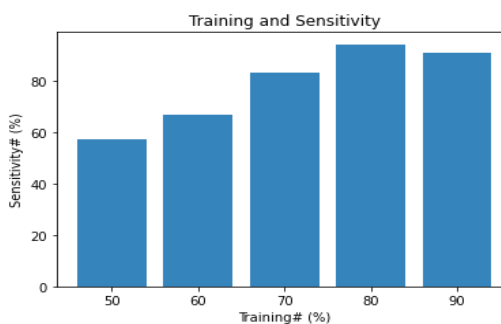
34.a



34.b



34.c



34.d

Fig. 34: Comparison of Performance Parameters Value.

4.5.4 Classification

After, feature extraction, for user suitability purpose, a GUI based system is developed which consisting of two buttons, namely, Select, and Detect. On pressing the Select button, the path of images stored images will be opened. Further, on pressing Detect button the *DRuNN* system shows the category of selected image.

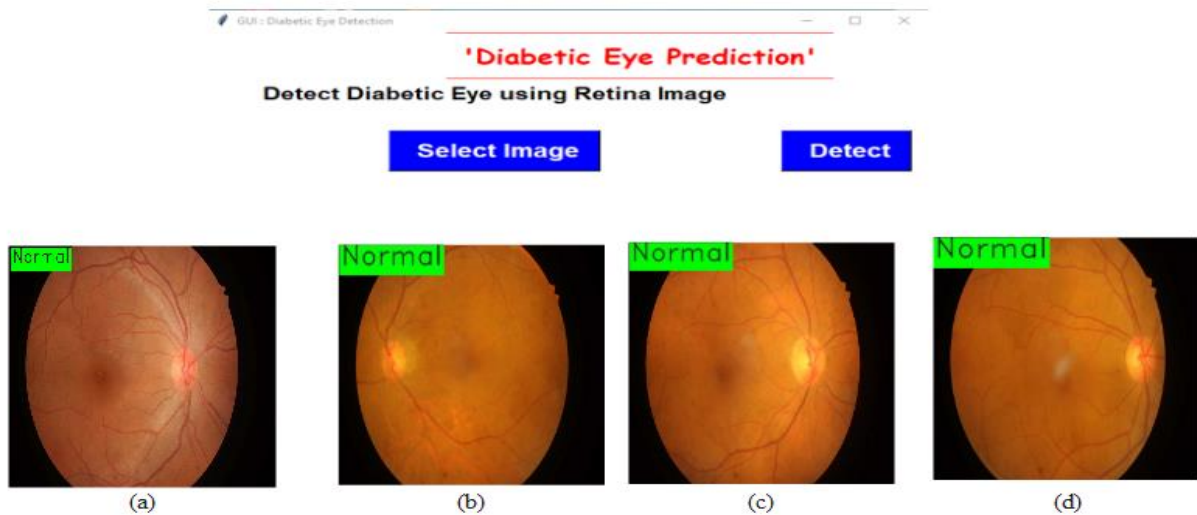


Fig. 35: Output window of GUI screen of Normal Scan

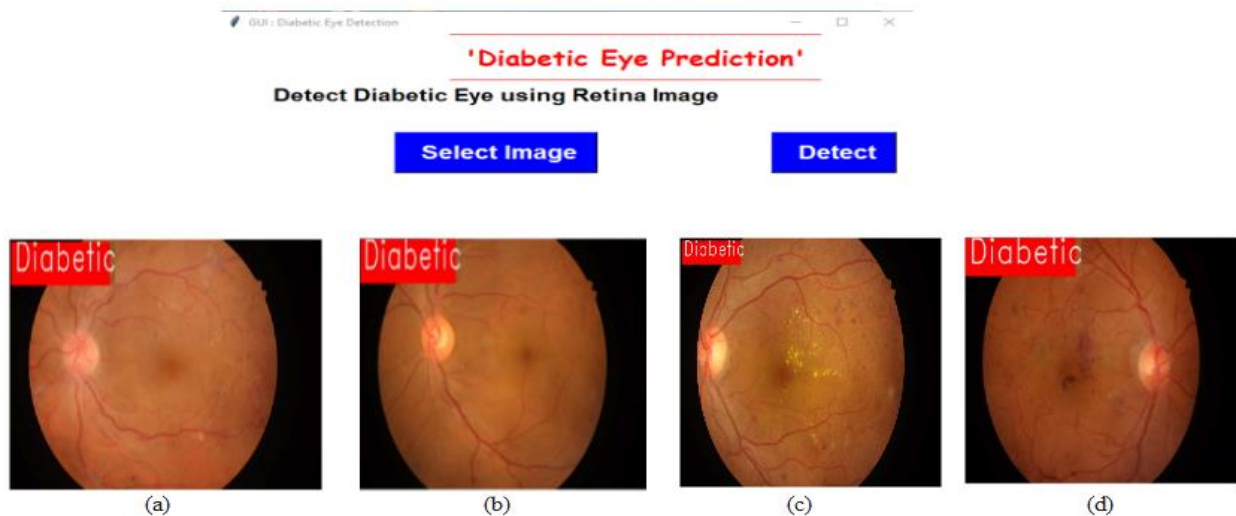
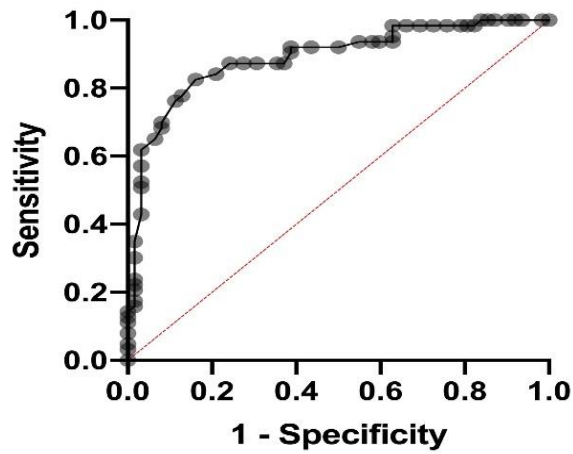


Fig. 36: Output window of GUI Screen of DR affected Patient

4.5.5 Receiver Operating Characteristic (ROC) Curve

For performance measurement for *DR* and normal retina images a *ROC* Curve is plotted (as shown in Graph. 2). The curve is plotted between True positive rate which is known as

Sensitivity and False positive rate which is known as 1- Specificity. *ROC* curve is plotted for testing purpose of usefulness of test cases. The region under the *ROC* curve is high which is very nearer to 1 which shows that the model can predict the images very efficiently and model is an excellent classifier.



Graph 3: *ROC* Curve

Hence, *ROC* curve tells about the usefulness of test cases and as mention earlier the area under the *ROC* curve *i.e* *AUC* is very large which shows that the proposed *DRuNN* technique is quite efficient.

CHAPTER 5

(CONCLUSION)

5.1 Advantages

- Reduce the burden on doctors.
- Detect the *DR* within some few seconds.
- Proper Feature learning.
- Good accuracy as well as other performance parameters values.
- Can be accepted for clinician perspective.

5.2 Conclusion

The developed *DRuNN* model helps to detect the *OCT* based scan is *DR* affected patients. If a person is suffering from blurred vision, floating spot in eyes and diabetes, the physician will advise the patient to take the *OCT* scan and if the *OCT* scan consists the red color dots and yellowish-white color physician will take the decision that the *OCT* scan is of *DR* affected patient. So in this proposed research work, *DRuNN* technique is developed which detects whether a person is suffering from *DR* or not. With the help of *CNN* and *ReLU*, the model helps to understand the curve and color based feature of the scan. The achieved accuracy, error, specificity, and sensitivity of *DRuNN* techniques are 90.00%, 10.00%, 86.67%, and 94.28% respectively when 80% data is reserved for training phase and rest 20% data is reserved for testing phase.

If a person has their own *OCT* scan then it can be detected in a few seconds that a person is suffering from *DR* or not. Thus, for user suitability purpose a *GUI* based system is developed which opens a *GUI* based window. The proposed research contribution definitively detects whether the given *OCT* scan with an efficient approach for finding *DR* affected persons within some few seconds. The proposed *DRuNN* maintains the complexity of the model, helps in feature scaling of every layer and provides reliable efficiency. Hence, this system is also beneficial for clinicians perspective.

5.3 Future Work

In future, to extend this proposed model which can also detect other eyes related disease and convert this proposed model into platform independent model.

REFERENCES

1. S. Wild, G. Roglic, A. Green, R. Sicree, H. King, Global prevalence of diabetes: estimates for the year 2000 and projections for 2030, *Diabetes care* 27 (2004) 1047-1053.
2. V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, et al., Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs, *Jama* 316 (2016) 2402-2410.
3. D. Doshi, A. Shenoy, D. Sidhpura, P. Gharpure, Diabetic retinopathy detection using deep convolutional neural networks, in: 2016 International Conference on Computing, Analytics and Security Trends (CAST), IEEE, 2016, pp. 261-266.
4. H. Pratt, F. Coenen, D. M. Broadbent, S. P. Harding, Y. Zheng, Convolutional neural networks for diabetic retinopathy, *Procedia Computer Science* 90 (2016) 200-205.
5. P. Prenta_si_c, S. Lon_cari_c, Detection of exudates in fundus photographs using deep neural networks and anatomical landmark detection fusion, *Computer methods and programs in biomedicine* 137 (2016) 281-292.
6. K. Bhatia, S. Arora, R. Tomar, Diagnosis of diabetic retinopathy using machine learning classification algorithm, in: 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), IEEE, 2016, pp. 347-351.
7. S. Somasundaram, P. Alli, A machine learning ensemble classifier for early prediction of diabetic retinopathy, *Journal of Medical Systems* 41 (2017) 201.
8. S. Masood, T. Luthra, H. Sundriyal, M. Ahmed, Identification of diabetic retinopathy in eye im-ages using transfer learning, in: 2017 International Conference on Computing, Communication and Automation (ICCCA), IEEE, 2017, pp. 1183-1187.
9. H. Takahashi, H. Tampo, Y. Arai, Y. Inoue, H. Kawashima, Applying artificial intelligence to disease staging: Deep learning for improved staging of diabetic retinopathy, *PloS one* 12 (2017) e0179790.
10. D. S. W. Ting, C. Y.-L. Cheung, G. Lim, G. S. W. Tan, N. D. Quang, A. Gan, H. Hamzah, R. Garcia Franco, I. Y. San Yeo, S. Y. Lee, et al., Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes, *Jama* 318 (2017) 2211-2223.
11. A. Gupta, R. Chhikara, Diabetic retinopathy: Present and past, *Procedia computer science* 132 (2018) 1432-1440.
12. P. M. Pawar, A. J. Agrawal, Retinal disease detection using machine learning techniques, *HELIX* 8 (2018) 3932-3937.
13. C. Lam, D. Yi, M. Guo, T. Lindsey, Automated detection of diabetic retinopathy using deep learning, *AMIA summits on translational science proceedings 2018* (2018) 147.
14. Sahlsten, J. Jaskari, J. Kivinen, L. Turunen, E. Jaanio, K. Hietala, K. Kaski, Deep learning fundus image analysis for diabetic retinopathy and macular edema grading, *Scientific reports* 9 (2019) 1-11.

15. P. Ruamviboonsuk, J. Krause, P. Chotcomwongse, R. Sayres, R. Raman, K. Widner, B. J. Campana, S. Phene, K. Hemarat, M. Tadarati, et al., Deep learning versus human graders for classifying diabetic retinopathy severity in a nationwide screening program, *NPJ digital medicine* 2 (2019) 1-9.
16. I. Qureshi, J. Ma, Q. Abbas, Recent development on detection methods for the diagnosis of diabetic retinopathy, *Symmetry* 11 (2019) 749.
17. Y.-H. Li, N.-N. Yeh, S.-J. Chen, Y.-C. Chung, Computer-assisted diagnosis for diabetic retinopathy based on fundus images using deep convolutional neural network, *Mobile Information Systems* 2019 (2019).
18. H. Abdelmotaal, W. Ibrahim, M. Sharaf, K. Abdelazeem, Causes and clinical impact of loss to follow-up in patients with proliferative diabetic retinopathy, *Journal of Ophthalmology* 2020 (2020).
19. T. R. Gadekallu, N. Khare, S. Bhattacharya, S. Singh, P. K. Reddy Maddikunta, I.-H. Ra, M. Alazab, Early detection of diabetic retinopathy using pca-rey based deep learning model, *Electronics* 9 (2020) 274.
20. M. Mateen, J. Wen, N. Nasrullah, S. Sun, S. Hayat, Exudate detection for diabetic retinopathy using pretrained convolutional neural networks, *Complexity* 2020 (2020).
21. T. Higuchi, N. Yoshifuji, T. Sakai, Y. Kitta, R. Takano, T. Ikegami, K. Taura, Clpy: A numpy compatible library accelerated with opencl, in: *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2019, pp. 933-940.
22. R. Chauhan, K. K. Ghanshala, R. Joshi, Convolutional neural network (cnn) for image detection and recognition, in: *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, IEEE, 2018, pp. 278-282.
23. J. Chang, J. Sha, An efficient implementation of 2d convolution in cnn, *IEICE Electronics Express* (2016) 13-20161134.
24. M. Ahmadi, S. Vakili, J. P. Langlois, W. Gross, Power reduction in cnn pooling layers with a preliminary partial computation strategy, in: *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)*, IEEE, 2018, pp. 125-129.
25. B. Ko, H.-G. Kim, K.-J. Oh, H.-J. Choi, Controlled dropout: A different approach to using dropout on deep neural network, in: *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, IEEE, 2017, pp. 358-362.
26. M. Li, H. Wang, J. Yang, Flattening and preferential attachment in the internet evolution, in: *2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2012, pp. 1-8.
27. G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700-4708.
28. S. C. Douglas, J. Yu, Whyrelu units sometimes die: Analysis of single-unit error backpropagation in neural networks, in: *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2018, pp. 864-868.

29. H. Qassim, A. Verma, D. Feinzimer, Compressed residual-vgg16 cnn model for big data places image recognition, in: 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2018, pp. 169-175.
30. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp.248-255.
31. S. Arya, R. Singh, A comparative study of cnn and alexnet for detection of disease in potato and mango leaf, in: 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), volume 1, IEEE, 2019, pp. 1-6.
32. Z. Zhang, Improved adam optimizer for deep neural networks, in: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), IEEE, 2018, pp. 1-2.
33. E. Dogo, O. Afolabi, N. Nwulu, B. Twala, C. Aigbavboa, A comparative analysis of gradient descent based optimization algorithms on convolutional neural networks, in: 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), IEEE, 2018, pp. 92-99.
34. S. N. Wuth, R. Coetzee, S. P. Levitt, Creating a python gui for a c++ image processing library, in: 2004 IEEE Africon. 7th Africon Conference in Africa (IEEE Cat. No. 04CH37590), volume 2, IEEE, 2004, pp. 1203-1206.

APPENDICES

Input : OCT Scan Dataset Training set γ_1 , Testing set γ_2

μ ---> the CNN learning rate

ξ ---> the iteration step $\xi \leftarrow 0(1+x)^p = 1 + \left(\frac{px}{1!}\right) + \left(\frac{p(p-1)x^2}{2!}\right) \dots$

ϵ ---> Maximum number of iteration in CNN

ϕ ---> images converted in one iteration

Output: q^* , CNN weights

Start;

1. Apply Mean Filter and Image Enhancement
 2. Govern base layers of the CNN i.e VGG-16;
 3. Set the head layers, $CNN_{dropout}$, $CNN_{pooling}$, CNN_{dense} , $CNN_{flatten}$
 4. Initialize the CNN parameters: μ , ϵ , ϕ
 5. Convert the image into 1-D array
 6. Train the CNN and Compute the weights
- for** $\xi = 1$ **to** ϵ **do**
7. Randomly select mini-batch of size: ϕ from γ_1
 8. Forward pass and compute the loss via μ
 9. Backward pass and update q^* with Adam Optimizer

end