IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Retinal Disease Detection

1Vrushali Prashant Sangar, 2Prof. U.A.Patil, 3Bhagyashri Tanaji Patil, 4Sejal Pramod Patil, 5Mrunal Uttam Desai

1Student, 2Head Of Department, 3Student, 4Student, 5Student

1Shivaji University Kolhapur,

2Shivaji University Kolhapur,

3Shivaji University Kolhapur,

4Shivaji University Kolhapur,

5Shivaji University Kolhapur

1.1 Introduction

Diabetes or diabetes mellitus is a metabolic disease in which the person's body produces an inadequate amount of insulin to produce high blood sugar. In India itself, more than 62 million people are suffering from diabetes. The people who are suffering from diabetes for more than 20 years has 80% chance of causing diabetic retinopathy.

According to the International Diabetes Federation, the number of adults with the diabetes in the world is estimated to be 366 million in 2011 and by 2030 this would have risen to 552 million. The number of people with type 2 diabetes is increasing in every country 80% of people with diabetes live in low-and middle-income countries. India stands first with 195% (18 million in 1995 to 54 million in 2025).

Previously, diabetes mellitus (DM) was considered to be present, largely, among the urban population in India. Recent studies clearly show an increasing prevalence in rural areas as well. Indian studies show a 3-fold increase in the presence of diabetes among the rural population over the last decade (2.2% in 1989 to 6.3% in 2003).

In India, Study shows the estimated prevalence of type 2 diabetes mellitus and diabetic retinopathy in a rural population of south India are nearly 1 of 10 individuals in rural south India, above the age of 40 years. Diabetic retinopathy is a state of eye infirmity in which damage arises due to diabetes mellitus. It is one of the prominent reasons behind blindness. The increased blood sugar due to diabetes incorporated damage to the tiny blood vessels in the retina thereby causing diabetic retinopathy. At least 90% of new cases could be reduced with proper medication as well as frequent monitoring of the eyes. It primarily affects the retinas of both the eyes, which can lead to vision loss if it is not treated. Poorly controlled blood sugars, high blood pressure, and high cholesterol increase the risk of developing Diabetic retinopathy.

The earlier work in the detection of varies stages DR based on explicit feature extraction & classification by using various Image Processing techniques & Machine learning algorithm respectively. Though high accuracy can be achieved using these methods but diagnosing diabetic retinopathy based on the explicit extraction of features is an intricate procedure. Due to development of Computer vision in recent times & availability of large dataset, it is now possible to use a deep Neural network for the detection & classification of Diabetic retinopathy.

They have used <Messidor Dataset= which consists of 300 images divided into three subsets. They have classified for NPDR phase detection.

1.2 Literature review

https://link.springer.com/chapter/10.1007/978-981-19-2840-6_58

The evaluation of the proposed approach is performed on the Ocular Disease Intelligent Recognition dataset. The obtained results showed that the RF and MLP classifiers achieved the highest accuracy of 77% in comparison to the other ML classifiers. On the other hand, the deep learning model (CNN model: Resnet152) provides an even better accuracy of 84% for the same task and dataset.

https://eandv.biomedcentral.com/articles/10.1186/s40662-020-00183-6

In clinical ophthalmology, a variety of image-related diagnostic techniques have begun to offer unprecedented insights into eye diseases based on morphological datasets with millions of data points.

1.3 Relevance

The system combines digital image processing and machine learning techniques to perform pattern recognition to classify diabetic retinal images.

The presence of exudates and lesions that are common in diabetic eye disorders, is detected using an approach that combines brightness adjustment procedure with statistical classification method and local feature-based verification strategy.

This would enable early detection of retinal diseases and people who require no further consultation can avoid expensive travel and testing. This benefits both the medical community and the people in rural areas.

With an enhanced user interface, the diagnostic tool can be operated by semi-skilled technicians addressing the problem of shortage of skilled medical professionals in rural areas.

1.4 Need of Present Work

In this project, we propose to develop a general classifier model that can distinguish healthy retinal images from diseased images. The proposed framework is deep learning that can automatically detect features at different levels from the training dataset of retinal images. Such a general model can be useful as a first level screening tool especially in rural settings.

This would enable early detection of retinal diseases and people who require no further consultation can avoid expensive travel and testing. This benefits both the medical community and the people in rural areas.

With an enhanced user interface, the diagnostic tool can be operated by semi-skilled technicians addressing the problem of shortage of skilled medical professionals in rural areas. The retinal images are captured by a fundus camera.

Although fundus cameras are still expensive, low-cost cameras are being developed which are now affordable. This is a one-time set-up cost which can benefit society at large.

The model can be enhanced to learn specific diseases and label the test images accordingly. This would involve getting a much larger and variety of dataset and training the model accordingly for multi-class labelling.

1.5 Objectives of this Work

The specific objectives of the project include:

- To enable early detection of retinal diseases by using this general model as a first level screening tool.
- To develop a system to classify the retinal fundus images as diseased or healthy.
- To avoid expensive travel and testing of the people who require no further consultation.
- The model can be enhanced to learn specific diseases and label the test images accordingly.

2.1 Problem Statement

In medical field, diagnosis of diseases competently carried out by using the image processing. Therefore, that to retrieve the relevant data from the amalgamation of resulting image is too difficult.

We use a deep learning model to automatically classify any retinal fundus image as healthy or diseased. Retinal fundus images are a valuable source of information for ophthalmologists to diagnose retina problems.

Early detection can improve chances of cure and also prevent blindness. Retinal problems like diabetic retinopathy, retinitis pigmentosa can be diagnosed using retinal fundus images by medical experts.

In recent times, machine learning research has focused on diagnosing diseases like diabetic retinopathy by extracting features and then classifying the image.
In this research our goal is to automatically classify images with retinal problems from those of the healthy ones without performing any explicit segmentation or feature extraction.

The image of normal and abnormal eye is given below:

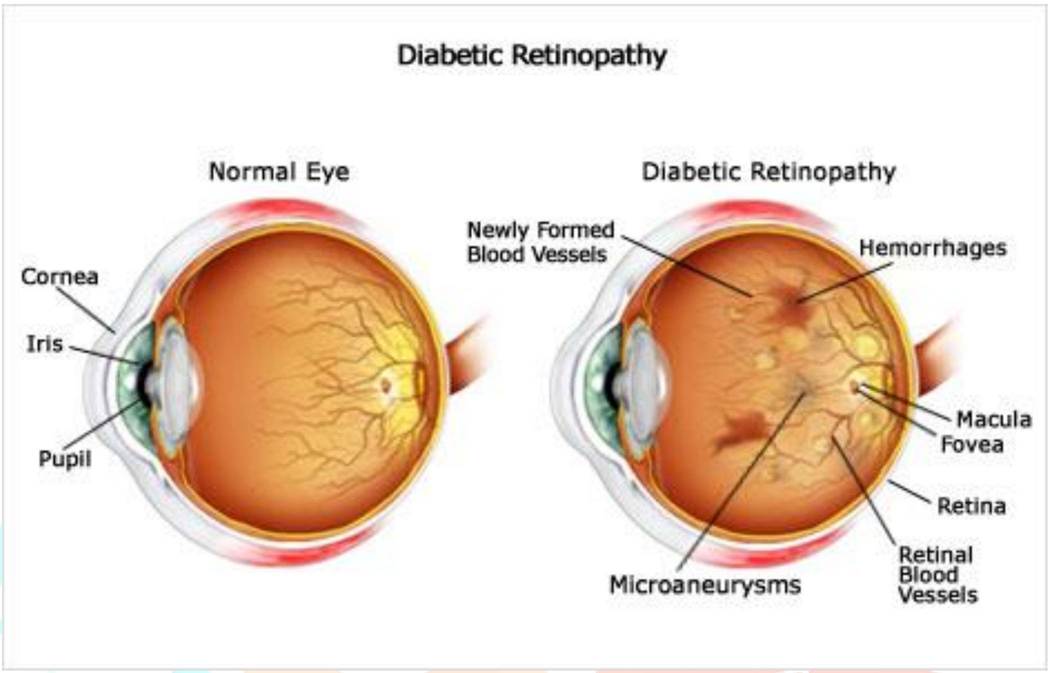

Figure 1: Normal and Abnormal Retinal Image

3.1 System Architecture

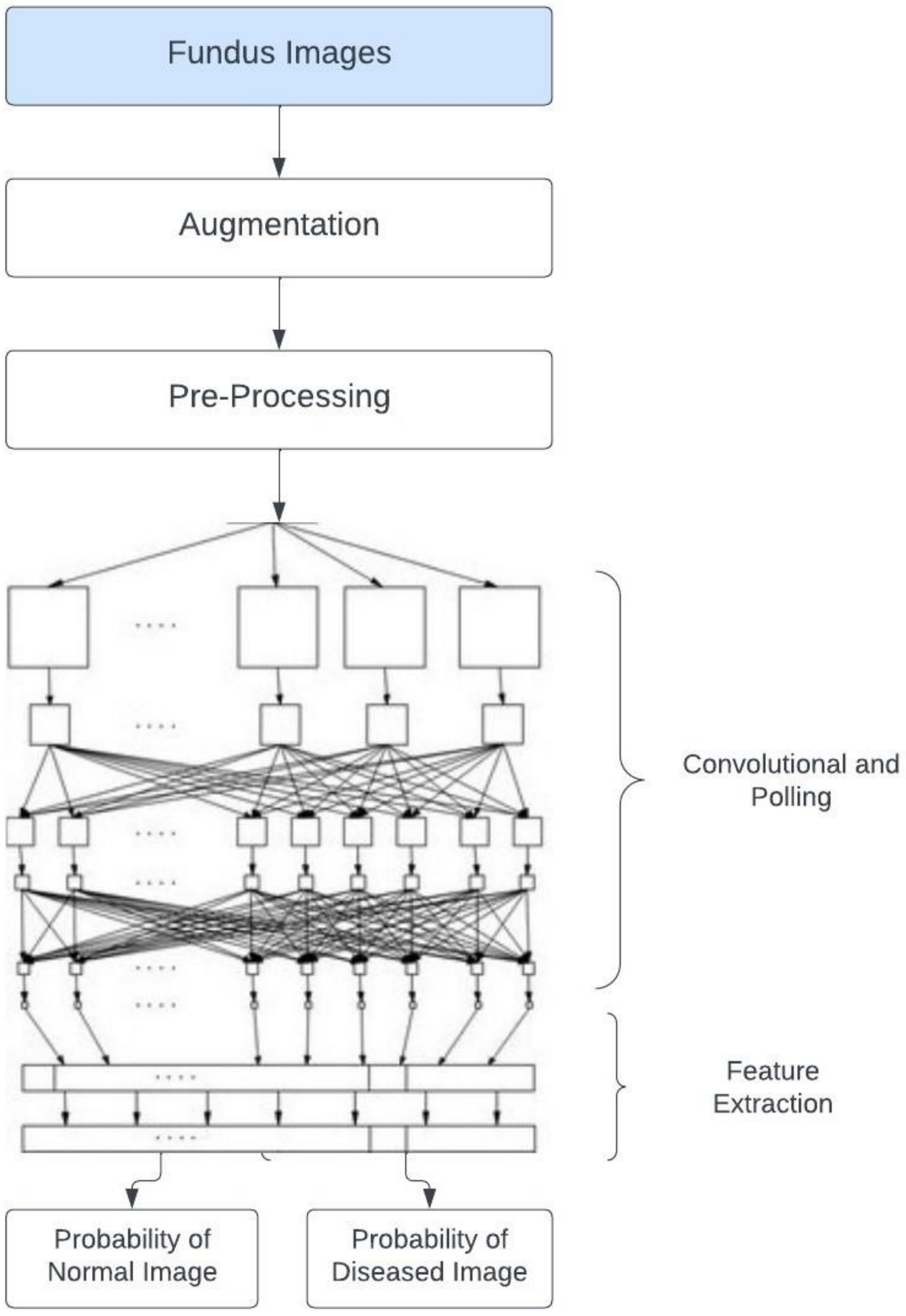
Fundus Images
Augmentation
Pre-Processing
Convolutional and Polling
Feature Extraction
Probability of Normal Image
Probability of Diseased Image

Figure 2: System Architecture Diagram

3.2 Flow Chart

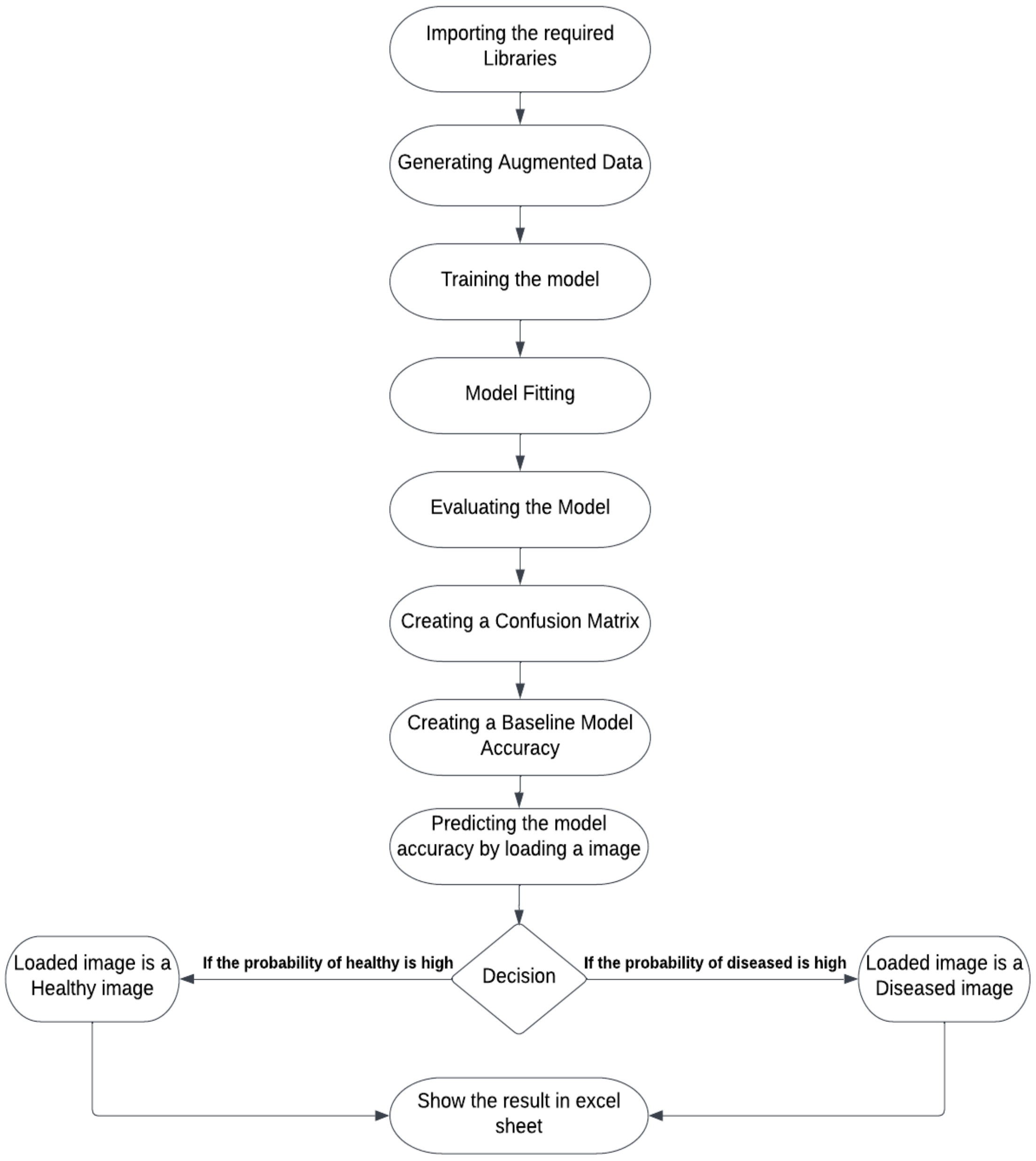
Importing the required Libraries
Generating Augmented Data
Training the model
Model Fitting
Evaluating the Model
Creating a Confusion Matrix
Creating a Baseline Model Accuracy
Predicting the model accuracy by loading a image
Decision
If the probability of healthy is high
Loaded image is a Healthy image
If the probability of diseased is high
Loaded image is a Diseased image
Show the result in excel sheet

Figure 3: Flow Chart

3.3 Data Flow Diagram

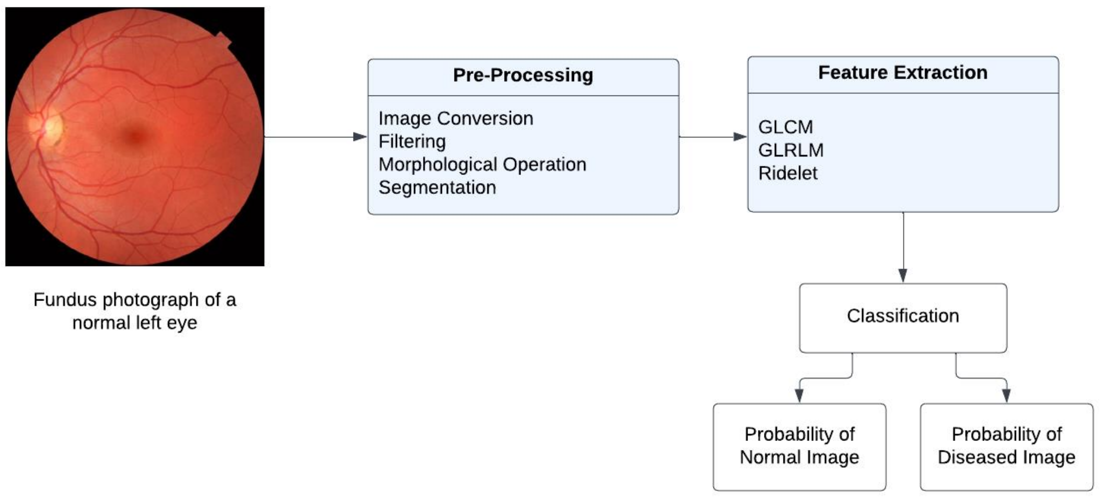

Figure 4: Data Flow Diagram

3.4 UML Diagram

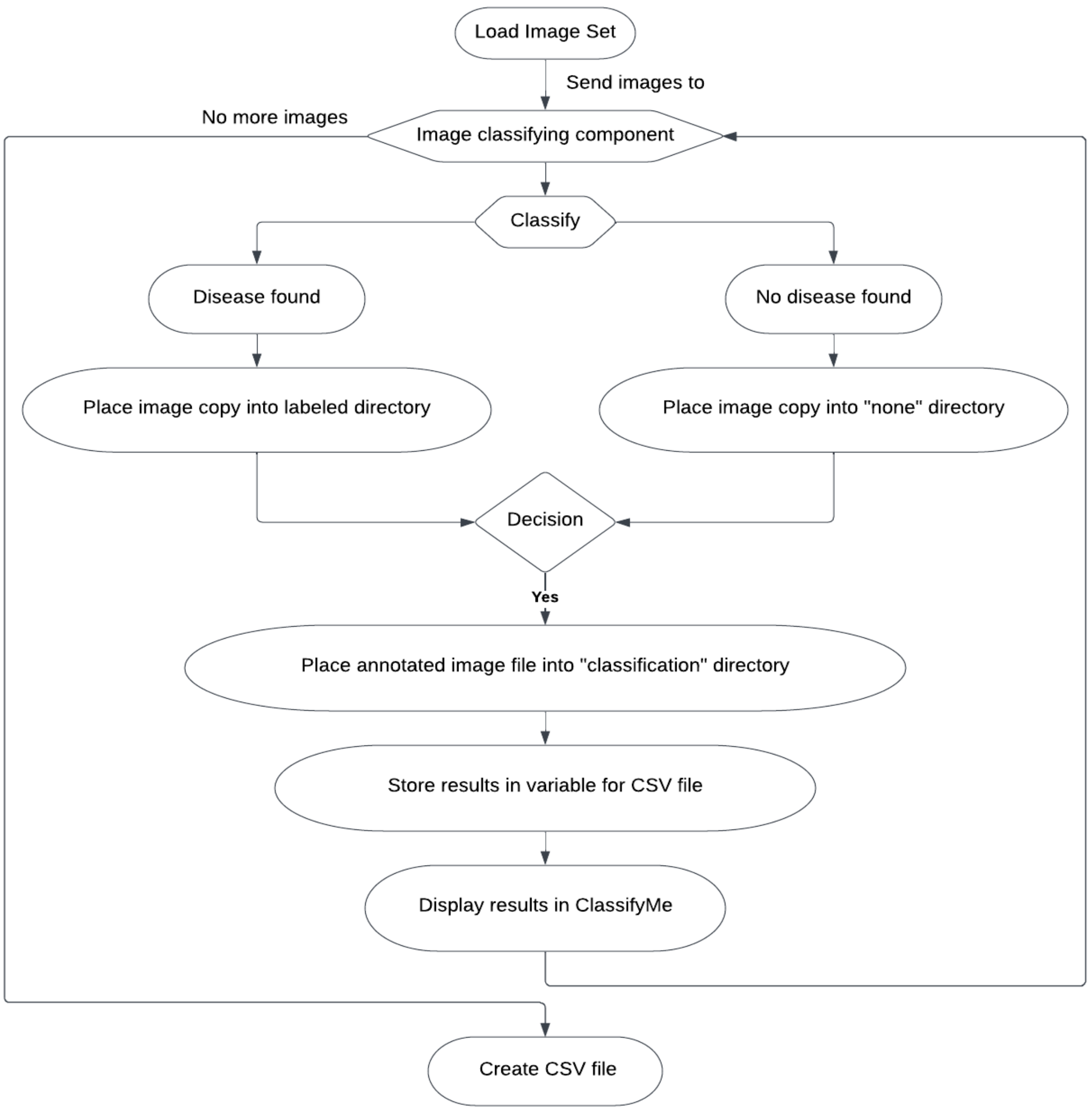
Load Image Set
Send images to
No more images
Image classifying component
Classify
Disease found
No disease found
Place image copy into labeled directory
Place image copy into "none" directory
Decision
Yes
Place annotated image file into "classification" directory
Store results in variable for CSV file
Display results in ClassifyMe
Create CSV file

Figure 5: UML Diagram

4.1 Technologies used

1. Anaconda:

Anaconda is a Python distribution (prebuilt and preconfigured collection of packages) that is commonly used for data science. The Anaconda distribution includes the Conda package manager in addition to the preconfigured Python packages and other tools. The Conda package manager can be used from the command line to set up Python environments and install additional packages that come with the default Anaconda distribution.

Anaconda Navigator is a GUI tool that is included in the Anaconda distribution and makes it easy to configure, install, and launch tools such as Jupyter Notebook. Although we use the Anaconda Navigator in this book, keep in mind that you can do everything through the command line using the conda command.

To begin the Anaconda Navigator installation process, visit https://www. anaconda.com and click the Downloads link at the top-right corner of the page.

2. Python:

Python is a general-purpose high level programming language that is widely used in data science and for producing deep learning algorithms. Deep learning demands a complex infrastructure of neural networks consisting of countless nodes all interacting together in various directions. Each node and its connections aren't complex on their own. In fact, because a single node does so little work compared to the entirety of the neural network, it's considered a relatively simple structure.

However, creating thousands of nodes easily rounds up to a lot of time and effort. The more complex the programming language you're using, the harder it would be to construct a working network.

Python, compared to other data-focused programming languages, is extremely easy to use and learn. After all, it's a high-level programming language, meaning it's closer to spoken human languages — English, in particular — than the available alternatives. Python's community of passionate users and learners all contribute to evolving the language by publishing in-depth tutorials and guidebooks online, as well as adding items to ready-use code libraries.

Additionally, the primary ingredient in all deep learning algorithms and applications is data, both as input and as training material. Python is mainly used for data managing manipulation and forecasting, making it an excellent tool to use for managing massive volumes of data for training your deep learning system, inserting input, or even making sense out of its output.

3. Jupyter Notebook:

Jupyter notebooks are documents that can be viewed and executed inside any modern web browser. Since you're reading this notebook, you already know how to view a Jupyter notebook. The next step is to learn how to execute computations that may be embedded in a Jupyter notebook.

To execute Python code in a notebook you will need access to a Python kernel. A kernel is simply a program that runs in the background, maintains workspace memory for variables and functions, and executes Python code. The kernel can be located on the same laptop as your web browser or located in an on-line cloud service.

The easiest way to use Jupyter notebooks is to sign up for a free or paid account on a cloud-based service such as Wakari.io or SageMathCloud. You will need continuous internet connectivity to access your work, but the advantages are there is no software to install or maintain. All you need is a modern web browser on your laptop, Chromebook, tablet or other device. Note that the free services are generally heavily oversubscribed, so you should consider a paid account to assure access during prime hours.

Implementations are available for Macs, Linux and Windows PC’s. The MacPython implementation, available for both OS9 and OSX Macs provides an excellent integrated development environment that in some ways is superior to IDLE. You can use your own stand- alone machine for any of the exercises that need only straight Python programming using the standard modules. You can also use your own machine for any exercises involving reading and writing of text data files.

Also, any Python extension modules that are written as ordinary human-readable Python scripts (e.g., phys.py) can be just downloaded and put in your python directory, regardless of what kind of machine you are using. However, compiled extension modules, with names like veclib.so need to be compatible with your specific hardware and Python implementation. In the rest of this workbook, when we say =Start up the Python interpreter,= the choice is up to you whether you use the simple command line interpreter or idle, or perhaps some other integrated Python development environment you might have (e.g., MacPython). For results that produce graphics, and for the use of idle, you must be connected to Python in a way that can display graphics on your screen (e.g. via a term). You won’t be reminded of this explicitly in the text. Exercises that don’t produce graphics can be done over any kind of link. =Write and run= a script could mean that you enter it using your favorite editor and run it from the command line.

4. TensorFlow:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019.

TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

TPU is a programmable AI accelerator designed to provide high throughput of low- precision arithmetic (e.g., 8-bit), and oriented toward using or running models rather than training them. Google announced they had been running TPUs inside their data centers for more than a year, and had found them to deliver an order of magnitude better- optimized performance per watt for machine learning. The second-generation TPUs deliver up to 180 teraflops of performance, and when organized into clusters of 64 TPUs, provide up to 11.5 petaflops.

In this project, we are using TensorFlow of version 2.11.0 which includes a new utility function: keras.utils.warmstart_embedding_matrix. It lets you initialize embedding vectors for a new vocabulary from another set of embedding vectors, usually trained on a previous run. The latest version of TensorFlow is 2.12.0. To download the TensorFlow libraries !pip install tensorflow is used in Anaconda Jupyter notebooks.

5. Keras:

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. TensorFlow Keras is an implementation of the Keras API that uses TensorFlow as a backend.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

In this project, we have used keras of version 2.11.0 which supports almost all the models of a neural network – fully connected, convolutional, pooling, recurrent, embedding, etc.

4.2 Algorithm

CNN Algorithm: In the fast-paced world of computer vision and image processing, one problem consistently stands out: the ability to effectively recognize and classify images. As we continue to digitize and automate our world, the demand for systems that can understand and interpret visual data is growing at an unprecedented rate. The challenge is not just about recognizing images – it's about doing so accurately and efficiently. Traditional machine learning methods often fall short, struggling to handle the complexity and high dimensionality of image data. This is where Convolutional Neural Networks (CNNs) comes to rescue. Convolutional Neural Networks, commonly referred to as CNNs, are a specialized kind of neural network architecture that is designed to process data with a grid-like topology. This makes them particularly well-suited for dealing with spatial and temporal data, like images and videos, that maintain a high degree of correlation between adjacent elements.

CNN Architecture has various types of architectures and models. One of the most important is VGGNet architecture.

VGG16 Architecture: VGG16 refers to the VGG model, also called VGGNet. It is a convolution neural network (CNN) model supporting 16 layers. The VGG16 model can achieve a test accuracy of 92.7% in ImageNet, a dataset containing more than 14 million training images across 1000 object classes.

VGG16, as its name suggests, is a 16-layer deep neural network. VGG16 is thus a relatively extensive network with a total of 138 million parameters—it's huge even by today's standards. However, the simplicity of the VGGNet16 architecture is its main attraction. The VGGNet architecture incorporates the most important convolution neural network features.

A VGG network consists of small convolution filters. VGG16 has three fully connected layers and 13 convolutional layers. Outline of the VGG architecture:

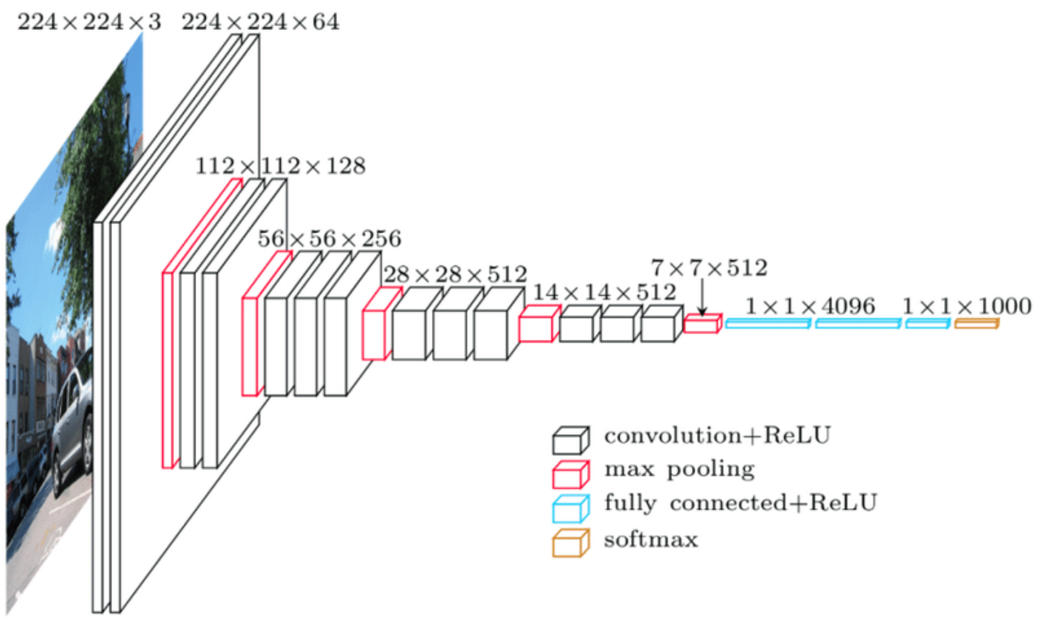

Figure 6: VGG16 Architecture

- **Input—**VGGNet receives a 224×224 image input. In the ImageNet competition, the model's creators kept the image input size constant by cropping a 224×224 section from the centre of each image.
- **Convolutional layers—**the convolutional filters of VGG use the smallest possible receptive field of 3×3. VGG also uses a 1×1 convolution filter as the input's linear transformation.
- **ReLu activation**—next is the Rectified Linear Unit Activation Function (ReLU) component. ReLU is a linear function that provides a matching output for positive inputs and outputs zero for negative inputs. VGG has a set convolution stride of 1 pixel to preserve the spatial resolution after convolution (the stride value reflects how many pixels the filter "moves" to cover the entire space of the image).
- **Hidden layers—**all the VGG network's hidden layers use ReLU instead of Local Response Normalization like AlexNet. The latter increases training time and memory consumption with little improvement to overall accuracy.
- **Pooling layers–**A pooling layer follows several convolutional layers—this helps reduce the dimensionality and the number of parameters of the feature maps created by each convolution step. Pooling is crucial given the rapid growth of the number of available filters from 64 to 128, 256, and eventually 512 in the final layers.
- **Fully connected layers—**VGGNet includes three fully connected layers. The first two layers each have 4096 channels, and the third layer has 1000 channels, one for every class.

4.3 Dataset

The Dataset which is used in this project was obtained from Friedrich-Alexander University machine learning data repository. The images obtained were 15 healthy images, 30 diseased retinal fundus images that included a mix of diabetic retinopathy and glaucomatous images. These images were then augmented to obtain 1021 diseased images and 522 healthy images. From this augmented source, 3 different data sets were created (a) Red free images of 128 resolution (b) Red free images of 1024 resolution (c) Color images of 1024 resolution. Each dataset is further divided into training, validation and test dataset. The test image set consisting of 100 images was run 10 times to get average accuracy.

Also, the other dataset which is used for testing was provided by the Kaggle coding website (https://www.kaggle.com/) and contains over 80,000 images, of approximately 6M pixels per image and scales of retinopathy. Resizing these im- ages and running our VGG16 on a high-end GPU, the NVIDIA K40c, meant we were able to train on the whole dataset. The NVIDIA K40c contains 2880 CUDA cores and comes with the NVIDIA CUDA Deep Neural Net- work library (cuDNN) for GPU learning. Through using this package around 15,000 images were uploaded on the GPU memory at any one time. The deep learning 4 packages Keras (http://keras.io/) was used with the Theano (http://deeplearning.net/software/theano/) machine learning back end. This was chosen due to good documentation and short calculation time. An image can be classified in 0.04 seconds meaning real-time feedback for the patient is possible.

The dataset consisted of male and female images of adults, both diseased and healthy images. Each of the above datasets was further divided into train, validation and test set using the scikit-learn library, with a random seed, to ensure similar split on multiple runs. The dataset was split as follows - 70% training and 20% validation and 10% testing.

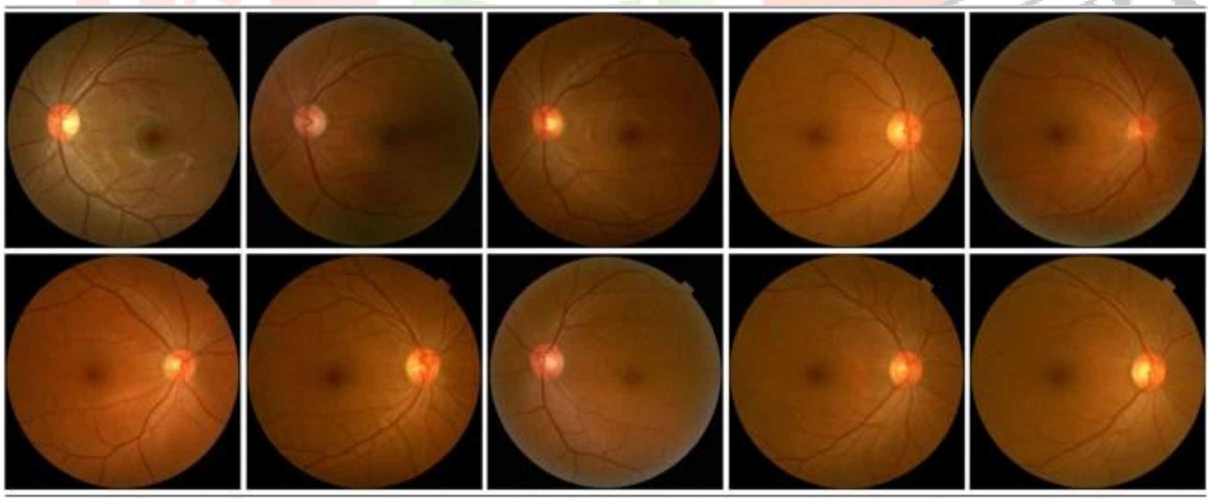

Figure 7: Example of Training Image Dataset

4.4 System Requirements

4.4.1 Software Requirements:

- Operating System - Windows
- Software - Anaconda
- Programming Language - Python

4.4.2 Hardware Requirements:

- Processor - Intel Pentium Dual Core i5 above
- RAM - 8GB
- Hard Disk - 200 GB

5.1 Experimental Setup

The dataset contained images from patients of varying ethnicity, age groups and extremely varied levels of lighting in the fundus photography. 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 using the OpenCV (http://opencv.org/) package. The images were also high resolution and therefore of significant memory size.

Following are the steps that have been implemented in the experimental setup of the project:

1. **Image Cropper:** Images from the hospital had some patient related information that had to be masked for ensuring confidentiality and privacy. A tool was developed for the benefit of the local hospital to ensure the same. A black box was drawn over the metadata in the image, and the pixels were masked. Fig 2 shows the masking done on a retinal image.

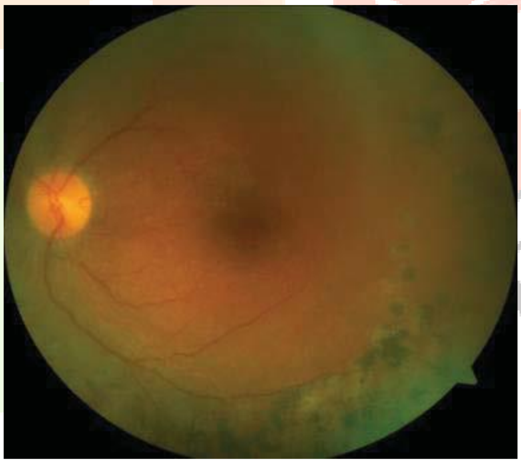

Figure 8: Cropped Image Without Patient Details

2. **Image Resizer:** The images were resized to 1024 by 1024 and 128 by 128 pixels from 2048 by 2048 for avoiding computational overhead and enable efficient training on the system. The results have been obtained for both resolutions.

3. **Image Augmentation:** The number of diseased images was more than the healthy images, using augmentation the number of healthy images was increased. The dataset was augmented by performing operations like rotation and flipping on existing images. The images were rotated at an angle of 90, 120, 180 and 270 degrees to generate new augmented images. The original pre-processed images were only used for training the network once. Afterwards, real-time data- augmentation was used throughout training to improve the localization ability of the network. During every epoch each image was

randomly augmented with: random rotation 0-90 degrees, random yes or no horizontal and vertical flips and random horizontal and vertical shifts. Fig 7 shows an example of augmentation.

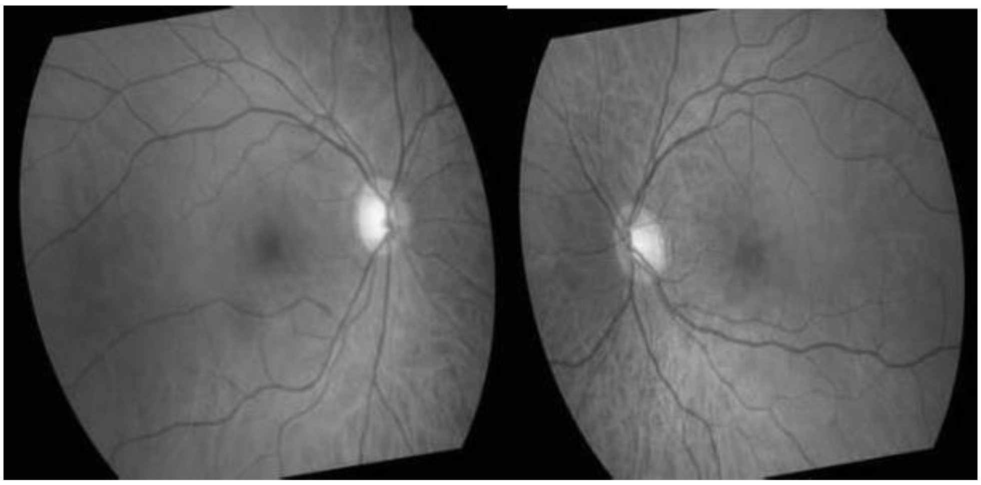

Figure 9: Original Image (Left), Augmented Image (Right), flipped

4. **Training using Deep CNN:** The images from the training dataset is first fed into the customized CNN that we refer to as VGG16. VGG16 architecture is given in Fig. 4. In most runs, VGG16 performed accurately in the first epoch . In some cases, it converged in the fifth epoch.
The VGG16 was initially pre-trained on 10,290 images until it reached a significant level. This was needed to achieve a relatively quick classification result without wasting substantial training time. After 120 epochs of training on the initial images the network was then trained on the full 78,000 training images for a further 20 epochs. Neural networks suffer from severe over-fitting especially in a dataset such as ours in which the majority of the images in the dataset are classified in one class, that showing no signs of retinopathy. To solve this issue, we implemented real-time class weights in the network.
For every batch loaded for back-propagation the class-weights were updated with a ratio respective to how many images in the training batch were classified as having no signs of DR. This reduced the risk of over-fitting to a certain class to be greatly reduced. The network was trained using stochastic gradient descent. A low learning rate of 0.0001 was used for 5 epochs to stabilize the weights. This was then increased to 0.0003 for the substantial 120 epochs of training on the initial 10,290 images, taking the accuracy of the model to over 60%, this took circa 350 hours of training. The network was then trained on the full training set of images with a low learning rate. Within a couple of large epochs of the full dataset the accuracy of the network had increased to over 70%. The learning rate was then lowered by a factor of 10 every time training loss and accuracy.

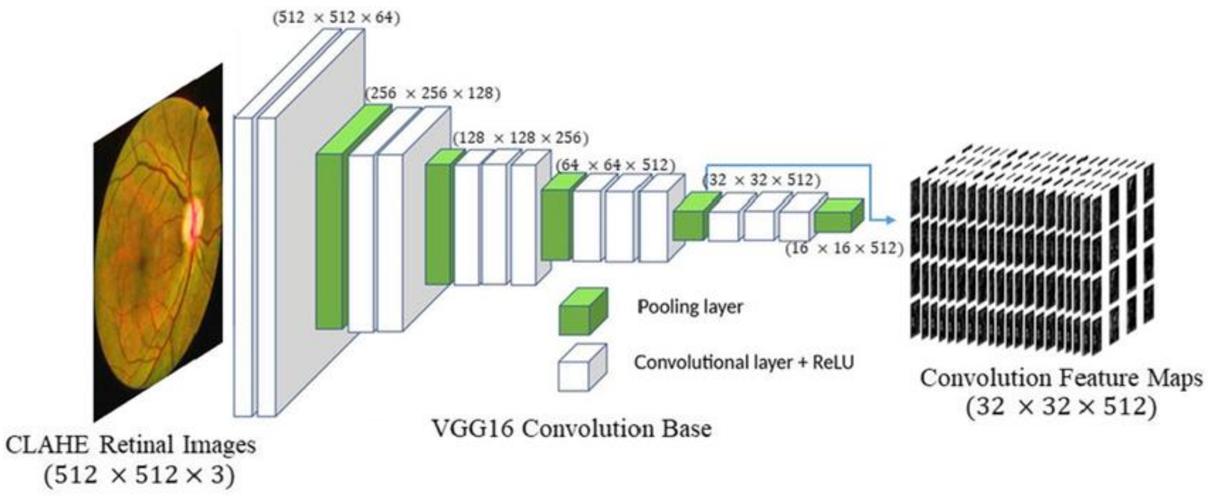

Figure 10: VGG16 Architecture for Retina

5. **Validation and hyperparameter setting:** The tuning of hyper parameters is critical to the performance of the network. The weights and biases assigned by the convolutional neural network are random and can vary during different runs of the algorithm. The network was validated using the validation dataset. On various iterations of the algorithm the batch size that was found to be ideal was 16, and the learning rate was set to a low (1/1000)value for optimum results.

6. **Testing and Prediction:** The test image dataset was a mix of healthy and diseased images. VGG16 trained on color, was tested on color image test set. VGG16 trained on red- free images was tested on red-free images. Each test set consisted of 100 images. There were 4 different test sets for each of the two data sources, namely 128 and 1024 pixels, both being red free and color images of 1024 pixels. The fourth test set consisted of images from the test set which was augmented by flipping . The output/prediction of the VGG16 was written to an excel file containing the probability score of the test image belonging to one of the two classes- Normal Retina or Diseased Retina. A suitable threshold of 0.85 was chosen for this binary classification problem. The threshold was chosen based on the performance of the model averaged for 10 runs. If the model outputs 0.85 or above for either class it belongs to that particular class, a score of 0.15 or lower means it does not belong to that particular class. The excel output file now contains the label of the class and the name of the test image.

Diseased Probability	Healthy Probability	Test Images
0.13826826	0.86173165	01_h.jpg
0.8162835	0.18371643	02_g.jpg
8.906861e-05	0.99991095	02_h.jpg
0.9999813	1.8711446e-05	05_dr.jpg
2.0403324e-05	0.9999796	05_h.jpg
0.99999106	8.897527e-06	03_dr.jpg
3.1388286e-20	0.9999608186	04_h.jpg
0.9999912086	8.665292e-28	01_dr.jpg

Table I: Table of images and their probability scores

5.2 Result

The results are shown and highlighted for various types of images in table I. Augmenting (by rotation and flipping) and re-testing shows no change in results. Accuracy for red-free images is higher than color images. Red-free images consist of only two colors hence it is easier to learn patterns across the fundus image. Accuracy for augmentation is higher than without augmentation as the dataset increases. The average accuracy over 10 runs for the dataset from the first data source is 96.52%. The average accuracy over 10 runs for the dataset from the second data source is 99.7%. The accuracy is same for both 1024- and 128-pixel resolution. Figure 5 shows results of the training and the validation phases.

Type	Metric	Friedrich Alexander University	
		Healthy	Diseased
COLOR- 1024 pixels	Precision, Recall	0.51, 0.86	0.53, 0.16
128 pixels red-free	Precision, Recall	0.50, 0.50	0.68, 0.50
1024 pixels red-free	Precision, Recall	0.57, 0.03	0.50, 0.98
With augmentation	Precision, Recall	0.79, 0.83	0.62, 0.56
	Average accuracy	70.1%	

Table II: Results as Accuracy, Precision, Recall

6.1 Conclusion

The goal of the project was to develop a system to classify the retinal fundus images as diseased or healthy. Using convolution neural networks, we have developed a system, that successfully performed this binary classification. Retinal fundus images were obtained from two sources, and a total of four datasets was created for testing. We found the average accuracy of 70.1% on those datasets. We also found that images with augmentation have better results than color, which is also the opinion of the medical community. The model can be enhanced to learn specific diseases and label the test images accordingly. This would involve getting a much larger and variety of dataset and training the model accordingly for multi-class labelling.

6.2 Discussion on Results

Among other existing supervising algorithms, most of them are requiring more pre-processing or post-processing stages for identifying the different stages of the diabetic retinopathy. Also, the other algorithms mandatorily requiring manual feature extraction stages to classify the fundus images. In our proposed solution, Deep Convolutional Neural Network (VGG16) is a wholesome approach to all level of diabetic retinopathy stages. No manual feature extraction stages are needed. Our network architecture with dropout techniques yielded significant classification accuracy. True positive rate (or recall) is also improved. This architecture has some setbacks which are: an additional stage of augmentation is needed for the images taken from different camera with different field of view. Also, our network architecture is complex and computation-intensive requiring high-level graphics processing unit to process the high-resolution images when the level of layers stacked more.

Table III: Healthy Augmented Images Prediction:

	Healthy Probability	**Diseased Probability**	**Class**
0	0.999095	5.40161E-25	Healthy
1	0.999105	5.41586E-12	Healthy
2	0.999813	5.57313E-13	Healthy
3	0.990105	1.42024E-10	Healthy
4	0.999685	4.7473E-15	Healthy

Table IV: Diseased Augmented Images Prediction:

	Diseased Probability	Healthy Probability	Class
0	0.991830	8.170286e-03	Diseased
1	0.816284	0.183716	Diseased
2	0.754272	0.245728	Diseased
3	1.000000	4.119241e-14	Diseased
4	0.999981	1.871145e-05	Diseased

6.3 Future Scope

Millions of people are affected by retinal abnormalities worldwide. Early detection and treatment of these abnormalities could arrest further progression, saving multitudes from avoidable blindness. Manual disease detection is time-consuming, tedious and lacks repeatability. There have been efforts to automate ocular disease detection, riding on the successes of the application of Convolutional Neural Networks (CNNs) and VGG16. These models have performed well, however, there remain challenges owing to the complex nature of retinal lesions. This work reviews the most common retinal pathologies, provides an overview of prevalent imaging modalities and presents a critical evaluation of current deep-learning research for the detection and grading of glaucoma, diabetic retinopathy, Age-Related Macular Degeneration and multiple retinal diseases. The work concluded that VGG16, through deep learning, will increasingly be vital as an assistive technology. As future work, there is a need to explore the potential impact of using ensemble CNN architectures in multiclass, multilabel tasks. Efforts should also be expended on the improvement of model explain ability to win the trust of clinicians and patients.

References

[1] Acharya, U. R., Kannathal, N., Ng, E. Y. K., Min, L. C., & Suri, J. S. (2006, August), “Computer-based classification of eye diseases” In Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE (pp. 6121-6124). IEEE.

[2] Gardner, G. G., Keating, D., Williamson, T. H., & Elliott, A. T. (1996), “Automatic detection of diabetic retinopathy using an artificial neural network: a screening tool” British journal of Ophthalmology, 80(11), 940-944.

[3] Winder, R. J., Morrow, P. J., McRitchie, I. N., Bailie, J. R., & Hart, P. M. (2009), “Algorithms for digital image processing in diabetic retinopathy” Computerized medical imaging and graphics, 33(8), 608-622.

[4] Bock, R., Meier, J., Michelson, G., Nyúl, L. G., & Hornegger, J. (2007, September), “Classifying glaucoma with image-based features from fundus photographs” In Joint Pattern Recognition Symposium (pp. 355-364). Springer, Berlin, Heidelberg.

[5] Wang, H., Hsu, W., Goh, K. G., & Lee, M. L. (2000). “An effective approach to detect lesions in color retinal images”. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on (Vol.2, pp.181-186). IEEE.

[6] Patton, N., Aslam, T. M., MacGillivray, T., Deary, I. J., Dhillon, B., Eikelboom, R. H & Constable, I. J. (2006). “Retinal image analysis: concepts, applications and potential”. Progress in retinal and eye research, 25(1), 99-127.

[7] Burlina, P. M., Joshi, N., Pekala, M., Pacheco, K. D., Freund, D. E., & Bressler, N. M. (2017). “Automated grading of age-related macular degeneration from color fundus images using deep convolutional neural networks”. JAMA ophthalmology, 135(11), 1170-1176.

[8] Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L & Dong, J. (2018). “Identifying medical diagnoses and treatable diseases by image-based deep learning”.

[9] “Neural Networks as computational devices” http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

[10] “Fundus Image Dataset - Friedrich Alexander University” https://www5.cs.fau.de/research/data/fundus-images/