International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 03 | Mar 2021 www.irjet.net p-ISSN: 2395-0072

Detection of Diabetic Retinopathy Using Neural Network

Aryan Kokane[1], Gourhari Sharma[1], Akash Raina[1], Shubham Narole[1], Prof. Pramila M. Chawan[2]

[1]B. Tech Student, Dept. of Computer Engineering and IT, VJTI College, Mumbai, Maharashtra, India
[2]Associate Professor, Dept. of Computer Engineering and IT, VJTI College, Mumbai, Maharashtra, India

---***---

Abstract- *Diabetic Retinopathy (DR) is a common eye disease which is contracted by diabetic people which may cause problems to the retina of the eye resulting in total or partial blindness. Therefore it is vital to identify diabetic retinopathy in the early stages. This paper aims to detect the diabetic retinopathy using neural networks and to give a decision about the presence of diabetic retinopathy. It will give us accuracy of our model.*

Key Words*: *Diabetic Retinopathy, Opencv, convolutional layer, fully connected layer, convolutional neural network, feature extraction

1. INTRODUCTION

Diabetic Retinopathy (DR) can cause loss of vision because the blood vessels within the retina leak fluid or hemorrhage (bleed), which may lead to a blurred or impaired vision. Diabetic retinopathy is diagnosed by recognizing anomalies on retinal images taken by fundoscopy. Because fundoscopic images are the primary method for diagnosis, going through them manually and analyzing those images are often time-consuming and unreliable, as the ability of detecting abnormalities varies. In this paper we aim to provide an automated, appropriate and sophisticated approach using convolution neural networks so that DR can be detected easily and the consequences can be minimized. This is an ongoing challenge problem on Kaggle which tries to develop a model for DR detection. Dataset is taken from the challenge-data part.

2. LITERATURE REVIEW

Harry Pratt et al. in this paper [1], proposed a convolutional neural network approach to diagnosing and classifying Diabetic Retinopathy from digital fundus images. They develop a network with CNN architecture and data augmentation which can identify the features involved in the classification task such as micro-aneurysms, exudate and haemorrhages on the retina. They train this network using a high-end graphics processor unit (GPU) on the publicly available Kaggle dataset and show good results, especially for a high-level classification task. On the data set of 80,000 images used their proposed CNN achieves a specificity of 95% and an accuracy of 75% on 5,000 validation images.

Athira et al. [2] proposed a R-CNN(Regional Convolutional Neural Network) to detect DR from digital fundus images. They implemented an approach where the image was segmented and only the regions of interest were taken for further processing. 10 layers were used for R-CNN, and trained on 130 fundus images and tested on 110 images. An accuracy of about 75.8% is obtained for R-CNN.

E. V. Carrera et al. in this paper [3] proposes a computer assisted diagnosis based on the digital processing of retinal image. This was performed on the Messidor dataset which consists of 1200 eye fundus color numerical images of the posterior pole captured by a special camera whose description is provided in their thesis. The objective of this was to detect any grade of DRNP. For that, they used 301 retinal images, 152 with grade 0 and 149 with grade 3. They trained a SVM classifier with all the features of these images and then tested it through a 10-fold cross-validation process. They achieved a sensitivity of 94.6% and a predictive capacity value of 93.8%.

Wong Li Yun et al. [4] used the K-means algorithm to extract features and Self-Organizing Map (SOM) classifier. The steps involved which are involved: Data Acquisition, K-means clustering for Feature Extraction and Self Organising Map for Classification. The classifier separates the cluster centroids of the normal class and the DR class. This model is only able to classify whether or not DR is present.

Salman Sayed et al. [5] used two models selected for detection of DR are Probabilistic Neural Network (PNN) and Support vector machines (SVM) and their results were analysed and compared. The DR has been classified into two categories NPDR and PDR with a sensitivity of 81.42% and specificity of 100%.

3. PROPOSED SYSTEM

3.1 PROBLEM STATEMENT

“To detect diabetic retinopathy using neural network”

3.2 DATASET DESCRIPTION

For this project, we have taken the dataset from Kaggle [6]. In this dataset we are provided with a large set of high-resolution retina images taken under a variety of imaging conditions. A left eye image and right eye is provided for every subject and images are labeled with a subject id as well as if they are left or right.

Diabetic retinopathy in each image is labeled on a scale of 0 to 4, according to the following scale:
0 - No DR
1 - Mild
2 - Moderate
3 - Severe
4 - Proliferative DR

Some images appear to one as they would see the retina through an anatomy in which the macula is on the left, optic nerve on the right for the right eye while others images appear as one would see through a microscope lens i.e. inverted. Like any real-world dataset, this dataset also contains noise in both the images and labels.

3.3 DATA PREPROCESSING

The dataset has images of patients of different age groups and extremely varied levels of brightness in the fundus photography. The pixel intensity values of different regions of the images are affected. To rectify this, colour normalisation is implemented. The images are high resolution so they have a large memory size. As the size is too large, RGB images are converted into gray scale images. The dataset was resized to 200x200 pixels which retained the important features. As our model is a Convolutional Neural Network, not much preprocessing is required.

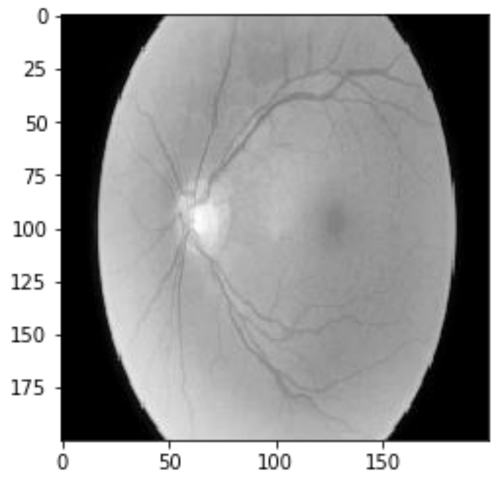


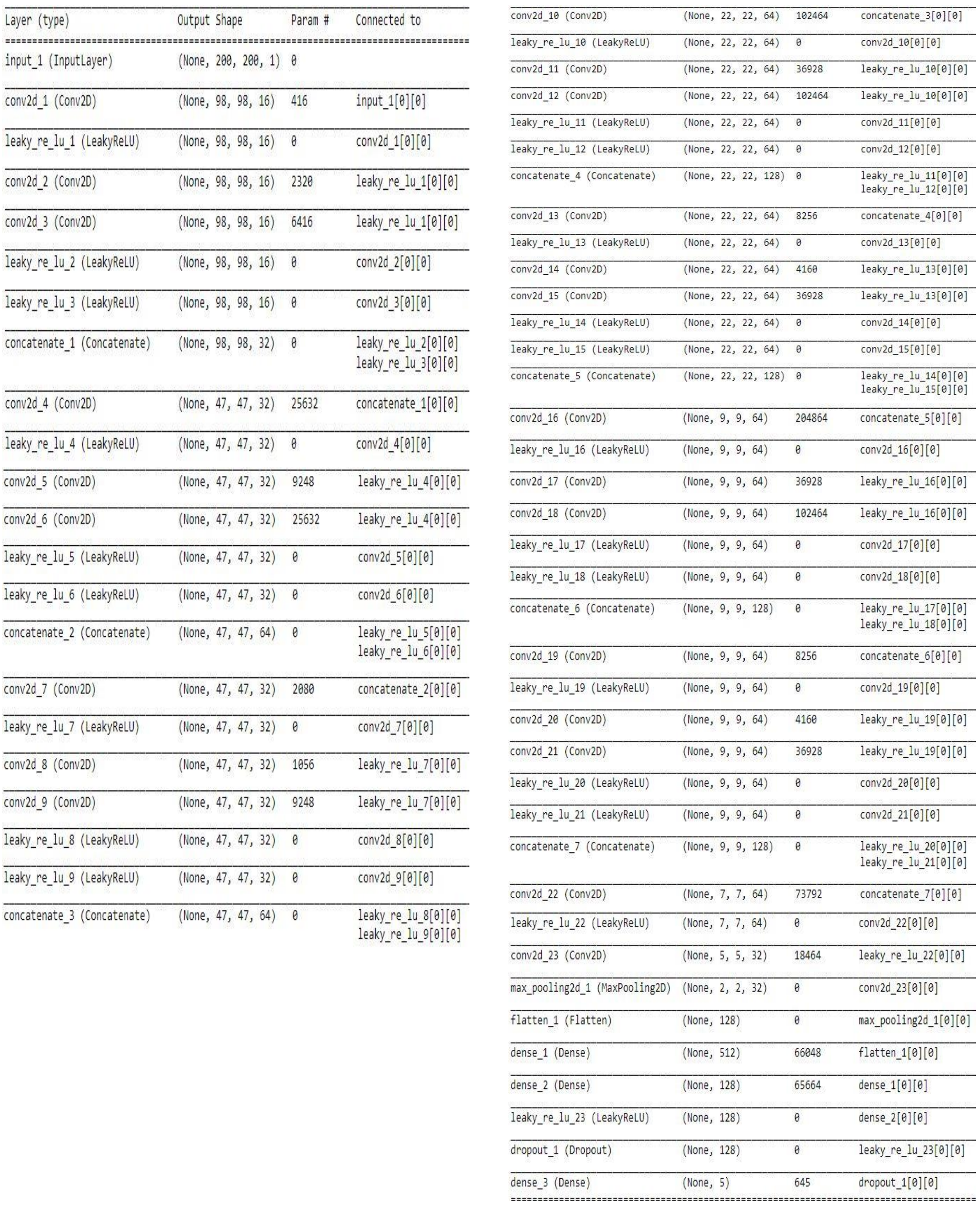
Fig - 1: Sample image after preprocessing

3.4 MODEL DESCRIPTION

A CNN model is used as proposed for the detection which is done by supervised learning by analyzing the features of training data [7]. We are proposing Convolutional Neural Network as it is suitable for computer vision because it consists of filters which lowers the necessary parameter storing which in turn reduces memory usage. It then uses pooling functions to compress the input image thus reducing memory. Also it is less complex because filters operate on limited size of input.

It is an adaptation of a multilayer perceptron and needs only less pre-processing techniques. The model comprises a number of convolutional layers, pooling layers, fully connected layers and Softmax. Convolutional layer is basically used for feature extraction followed by a ReLU activation function. The first convolutional layer concentrates on the low-level features, and with furthermore layers, the architecture adapts to the high-level features. The extracted features can be reduced in dimension as compared to the input, or the other in which the dimensionality is either increased or remains the same. This is done by applying valid padding or same padding. Max pooling is applied which helps reduce overfitting and reducing parameters. Max Pooling returns the maximum value from the region of the image covered by the kernel. Then, we are going to flatten the final output, i.e. to convert the multidimensional vector to a 1-dimensional vector, for classification purposes. Dropout layer is used near fully connected layer to prevent over creation of extra parameters. Some of the Dense layers have ReLU activation, or sigmoid activation or Softmax for classification. Over a series of epochs, the model is able to distinguish between important and low-level features in images and then classify them.

Batch size to train = 32
Number of output classes = 5
Number of convolutional filters to use = 32
Size of pooling area for max pooling = 2
Convolution kernel size = 3*3



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 200, 200, 1)  0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 98, 98, 16)   416         input_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_1 (LeakyReLU)       (None, 98, 98, 16)   0           conv2d_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 98, 98, 16)   2320        leaky_re_lu_1[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 98, 98, 16)   6416        leaky_re_lu_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_2 (LeakyReLU)       (None, 98, 98, 16)   0           conv2d_2[0][0]
__________________________________________________________________________________________________
leaky_re_lu_3 (LeakyReLU)       (None, 98, 98, 16)   0           conv2d_3[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 98, 98, 32)   0           leaky_re_lu_2[0][0]
                                                                 leaky_re_lu_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 47, 47, 32)   25632       concatenate_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_4 (LeakyReLU)       (None, 47, 47, 32)   0           conv2d_4[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 47, 47, 32)   9248        leaky_re_lu_4[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 47, 47, 32)   25632       leaky_re_lu_4[0][0]
__________________________________________________________________________________________________
leaky_re_lu_5 (LeakyReLU)       (None, 47, 47, 32)   0           conv2d_5[0][0]
__________________________________________________________________________________________________
leaky_re_lu_6 (LeakyReLU)       (None, 47, 47, 32)   0           conv2d_6[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 47, 47, 64)   0           leaky_re_lu_5[0][0]
                                                                 leaky_re_lu_6[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 47, 47, 32)   2080        concatenate_2[0][0]
__________________________________________________________________________________________________
leaky_re_lu_7 (LeakyReLU)       (None, 47, 47, 32)   0           conv2d_7[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 47, 47, 32)   1056        leaky_re_lu_7[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 47, 47, 32)   9248        leaky_re_lu_7[0][0]
__________________________________________________________________________________________________
leaky_re_lu_8 (LeakyReLU)       (None, 47, 47, 32)   0           conv2d_8[0][0]
__________________________________________________________________________________________________
leaky_re_lu_9 (LeakyReLU)       (None, 47, 47, 32)   0           conv2d_9[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 47, 47, 64)   0           leaky_re_lu_8[0][0]
                                                                 leaky_re_lu_9[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 22, 22, 64)   102464      concatenate_3[0][0]
__________________________________________________________________________________________________
leaky_re_lu_10 (LeakyReLU)      (None, 22, 22, 64)   0           conv2d_10[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 22, 22, 64)   36928       leaky_re_lu_10[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 22, 22, 64)   102464      leaky_re_lu_10[0][0]
__________________________________________________________________________________________________
leaky_re_lu_11 (LeakyReLU)      (None, 22, 22, 64)   0           conv2d_11[0][0]
__________________________________________________________________________________________________
leaky_re_lu_12 (LeakyReLU)      (None, 22, 22, 64)   0           conv2d_12[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 22, 22, 128)  0           leaky_re_lu_11[0][0]
                                                                 leaky_re_lu_12[0][0]
__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 22, 22, 64)   8256        concatenate_4[0][0]
__________________________________________________________________________________________________
leaky_re_lu_13 (LeakyReLU)      (None, 22, 22, 64)   0           conv2d_13[0][0]
__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 22, 22, 64)   4160        leaky_re_lu_13[0][0]
__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 22, 22, 64)   36928       leaky_re_lu_13[0][0]
__________________________________________________________________________________________________
leaky_re_lu_14 (LeakyReLU)      (None, 22, 22, 64)   0           conv2d_14[0][0]
__________________________________________________________________________________________________
leaky_re_lu_15 (LeakyReLU)      (None, 22, 22, 64)   0           conv2d_15[0][0]
__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 22, 22, 128)  0           leaky_re_lu_14[0][0]
                                                                 leaky_re_lu_15[0][0]
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 9, 9, 64)     204864      concatenate_5[0][0]
__________________________________________________________________________________________________
leaky_re_lu_16 (LeakyReLU)      (None, 9, 9, 64)     0           conv2d_16[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 9, 9, 64)     36928       leaky_re_lu_16[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 9, 9, 64)     102464      leaky_re_lu_16[0][0]
__________________________________________________________________________________________________
leaky_re_lu_17 (LeakyReLU)      (None, 9, 9, 64)     0           conv2d_17[0][0]
__________________________________________________________________________________________________
leaky_re_lu_18 (LeakyReLU)      (None, 9, 9, 64)     0           conv2d_18[0][0]
__________________________________________________________________________________________________
concatenate_6 (Concatenate)     (None, 9, 9, 128)    0           leaky_re_lu_17[0][0]
                                                                 leaky_re_lu_18[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 9, 9, 64)     8256        concatenate_6[0][0]
__________________________________________________________________________________________________
leaky_re_lu_19 (LeakyReLU)      (None, 9, 9, 64)     0           conv2d_19[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 9, 9, 64)     4160        leaky_re_lu_19[0][0]
__________________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, 9, 9, 64)     36928       leaky_re_lu_19[0][0]
__________________________________________________________________________________________________
leaky_re_lu_20 (LeakyReLU)      (None, 9, 9, 64)     0           conv2d_20[0][0]
__________________________________________________________________________________________________
leaky_re_lu_21 (LeakyReLU)      (None, 9, 9, 64)     0           conv2d_21[0][0]
__________________________________________________________________________________________________
concatenate_7 (Concatenate)     (None, 9, 9, 128)    0           leaky_re_lu_20[0][0]
                                                                 leaky_re_lu_21[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 7, 7, 64)     73792       concatenate_7[0][0]
__________________________________________________________________________________________________
leaky_re_lu_22 (LeakyReLU)      (None, 7, 7, 64)     0           conv2d_22[0][0]
__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 5, 5, 32)     18464       leaky_re_lu_22[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 2, 2, 32)     0           conv2d_23[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 128)          0           max_pooling2d_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 512)          66048       flatten_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 128)          65664       dense_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_23 (LeakyReLU)      (None, 128)          0           dense_2[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           leaky_re_lu_23[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 5)            645         dropout_1[0][0]
==================================================================================================
```

Fig 2: Model Summary

Total params: 991,461
Trainable params: 991,461
Non-trainable params: 0

4. IMPLEMENTATION

The project is run on a system with Windows 10 with a configuration of Core™ i7 Processor Model and a NVIDIA GeForce GTX 1050 Graphics Controller .We have used python as our programming language throughout this project and performed it on Jupyter notebook. There are a number of libraries available in python which we have used such as: NumPy, Pandas, PIL, Keras.

NumPy : Used to read the dataset images as a NumPy array ndarray and the various image processing were performed using NumPy functions.
PIL : Used for opening, manipulating, and saving many different images present in our database.
Pandas : To read and convert csv files to pandas dataframe.
Keras : To build the neural network

As suggested by More, Nikhil T. et al. [8], the requirements that have been defined for the project influence the rate of success of the project.

5. RESULTS

We have gone for a 80:20 split, i.e 4000 training samples and 1000 test samples. Data augmentation is performed. An accuracy of 74.8% is achieved.

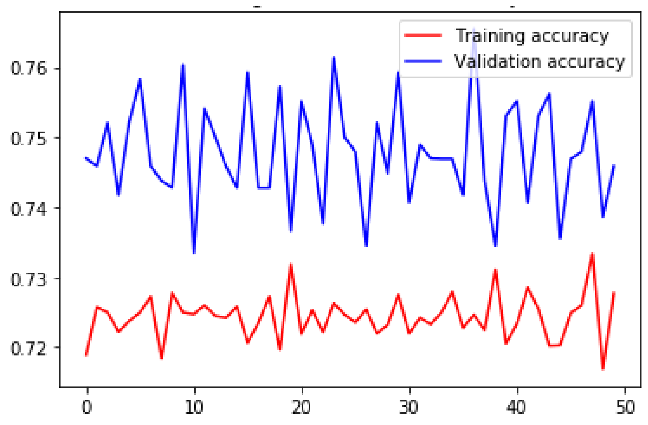


Fig. 3: Training and Validation accuracy over 50 epochs

6. CONCLUSION AND FUTURE SCOPE

As we can see from our results, it is quite encouraging with an accuracy of 74.8% and is comparable with other models out there. Future scope would be modifications such as increase size of test and training data set, or to use an improved algorithm to get clearer features or in place of direct image features try to use gradient of RGB layers could improve efficiency and results. Acquiring acceptable accuracy can lead the people in early detection of retinopathy.

7. REFERENCES

[1] Harry Pratt, Frans Coenen, Deborah M Broadbent, Simon P Harding, Yalin Zheng,"Convolutional Neural Networks for Diabetic Retinopathy" , International Conference On Medical Imaging Understanding and Analysis 2016, MIUA 2016,6-8 July 2016, Loughborough, UK.

[2] Athira T R, Athira Sivadas, Aleena George, Amala Paul, Neethu Radha Gopan, "Automatic detection of Diabetic Retinopathy using R-CNN", International Research Journal of Engineering and Technology (IRJET), May 2019.

[3] E. V. Carrera, A. González and R. Carrera, "Automated detection of diabetic retinopathy using SVM," 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON), Cusco, Peru, 2017, pp. 1-4, doi: 10.1109/INTERCON.2017.8079692.

[4] Wong Li Yun and Muthu Rama Krishnan Mookiah Department of Electronics and Computer Engineering, Ngee Ann Polytechnic, Singapore 599489 , "Detection of Diabetic Retinopathy Using K-Means Clustering and Self-Organizing Map "

[5] Salman Sayed, Dr. Vandana Inamdar , Sangram Kapre , "Detection of Diabetic Retinopathy using Image Processing and Machine Learning"

[6] Kaggle Challenge Diabetic Retinopathy Detection. https://www.kaggle.com/c/diabetic-retinopathydetection

[7] Aryan Kokane, Gourhari Sharma, Akash Raina, Shubham Narole, Pramila M. Chawan, "Detection of Diabetic Retinopathy using Machine Learning", International Research Journal of Engineering and Technology (IRJET) Volume 7, Issue 11, November 2020.

[8] More, Nikhil T.; Sapre, Bhushan S.; and Chawan, Pramila M. (2017) "An Insight into the Importance of Requirements Engineering," *International Journal of Computer and Communication Technology*: Vol. 8: Iss. 1, Article 5. DOI: 10.47893/IJCCT.2017.1394