Blood Vessel Segmentation in Retinal Fundus Images for Proliferative Diabetic Retinopathy Screening Using Deep Learning

Abstract-
Diabetic retinopathy (DR) is also called diabetic eye disease, which causes damage to the retina due to diabetes mellitus and that leads to blindness when the disease reaches an extreme stage. The medical tests take a lot of procedure, time, and money to test for the proliferative stage of diabetic retinopathy. Hence to resolve this problem, this model is proposed to detect and identify the proliferative stages of diabetic retinopathy which is also identified by its hallmark feature that is neovascularization. In the proposed system, the paper aims to correctly identify the presence of neovascularization using color fundus images. The presence of neovascularization in an eye is an indication that the eye is affected with proliferative diabetic retinopathy. Neovascularization is the development of new pathogenic blood vessels in the eye. The occurrence of neovascularization may lead to partial or complete vision loss. Hence timely and accurate prediction is important. The method was proposed to detect the presence of neovascularization which involves image processing methods such as resizing, green channel filtering, Gaussian filter, and morphology techniques such as erosion and dilation. Threshold and contouring have been used to segment the blood vessels from the rest of the features of the retinal fundus image. For classification, the different layers of CNN have been used and modeled together in a VGGnet architecture. The model was trained and tested on 2200 images all together from the Kaggle database. The model was tested on two popular public data sets DRIVE and STARE. The accuracy, specificity, sensitivity, precision, F1-score was achieved as 0.96, 0.99, 0.95, 0.99, 0.97 respectively on DRIVE and 0.95, 0.99, 0.9375, 0.96, 0.95 respectively on STARE.

Keywords-
Proliferative diabetic retinopathy, blood vessels, neovascularization, vision loss, pathogenic blood vessels, dense-net, CNN.

1. Introduction

Diabetic retinopathy (DR) is an eye disease that is caused by people who are affected by diabetes. Diabetic retinopathy is one of the most leading chronic diseases that leads to preventable blindness [1]. Diabetic retinopathy is an age-related disease that affects the retina [2]. Diabetic retinopathy is mainly classified into two categories: Non-proliferative diabetic retinopathy (NPDR) and Proliferative diabetic retinopathy (PDR). There are four stages of diabetic retinopathy. These are classified as no DR, mild, moderate, severe, and proliferative [3]. The NPDR consists of mild, moderate, and severe stages while the PDR consists of the proliferative stage only. Fig 1(a) shows the retinal image of a normal eye while Fig 1(b) shows the retinal image of a proliferative eye.

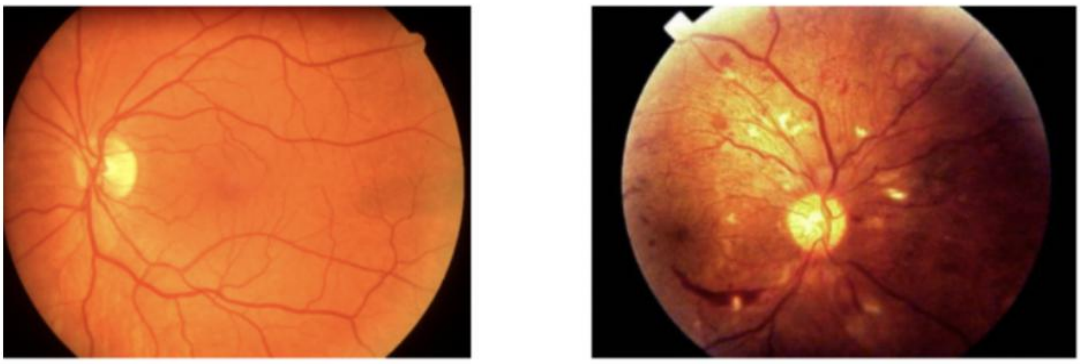

Fig 1. (a) Normal Eye, (b) Proliferative diabetic retinopathy affected eye

The proliferative stage of DR completely deals with the retinal blood vessel. The retinal blood vessels are the main feature for the detection of the proliferative stage. Neovascularization is the growth of tiny pathogenic blood vessels in the retina [4] that occur in the vitreoretinal interface [5]. Neovascularization is a special feature that is present only in the proliferative stage and distinguishes this stage from non- proliferative stages. In the proposed system, the model concentrates upon the detection of the stage of the eye whether the eye is in the proliferative stage or non-proliferative stage. The model goes through architecture with preprocessing, vessel segmentation, removal of extra features, classification, and prediction.

For preprocessing different techniques have been used such as resizing, green channel, gaussian blur, CLAHE, and morphological structuring. A. Lagae et. al [6] has used the Gabor filter for denoising. In this system, the Gaussian filter is used. A comparative analysis of using both filters is given in this paper. CLAHE is widely used for image enhancement. Y. Chang et. al [7] has shown the effect of using CLAHE. For vessel segmentation the contouring method has been used to extract the blood vessel line. The removal of extra features not required which was extracted during the segmentation process is done in the next step. The feature extraction and classification are done using CNN. Many researchers have used VGGnet architecture. The work of X. Liu et. al [8] supports VGGnet as for CNN training and classification. The module is a prediction where the confusion matrix is generated. The training file is saved in an HDF5 file and used in the prediction module. The results show the comparison and results of different filters and architecture used supported by graphs. The proposed model is tested on public databases such as the DRIVE data set [9] and the STARE data set [10].

Organization - SRM Institute of Science and Technology

2. Related works

Blood vessel segmentation and neovascularization are the two main areas of research related to proliferative diabetic retinopathy. Yu et al. (2018) [1] developed a for detection of neovascularization on the optic disc. For this, a local phase symmetry algorithm was used to separate the optic disc from the rest of the eye. Further the green channel was used for preprocessing the image and Gabor filter was used for the vessel segmentation. The feature extraction was done in three steps that are vessel morphological feature extraction, texture extraction, and feature selection. The Support Vector Machine (SVM) classifier is used to detect the presence of neovascularization. The overall model was trained and tested on 424 images and achieved an accuracy of 0.9523, specificity of 0.963, and sensitivity of 0.929. The data set used was the Kaggle DR data set.

Thangaraj et al. (2018) [11] developed a model for the segmentation of retinal blood vessels using neural networks. This method was tested on three most used public data sets that are DRIVE, STARE, and CHASE DB1. This model follows a procedure of pre-processing that involves central light reflex removal, background homogenization, and border extension. For background homogenization, the CLAHE technique has been used. The border extension has used the FOV-mask and Region of Interest (ROI) methods. For vessel segmentation or feature extraction, the Gabor filter response feature and Hessian filter response features are used. The neural network was used for the training and classification of the data. The overall model was tested on three publicly used data sets DRIVE, STARE, and CHASE DB1. The accuracy obtained on DRIVE, STARE, and CHASE DB1 is 0.9606, 0.9435, and 0.9468. The sensitivity obtained on DRIVE, STARE and CHASE DB1 is 0.8014, 0.8339, and 0.6288.

A method was developed by Lin et al. (2019) [12] for the automatic detection of the retinal blood vessel. The supervised network used the holistically- nested edge detector (HED) architecture where VGGnet is used as a pre-trained network. The model uses FCN as a deeply supervised network and CRF for a regularized network. CRF is used for pixel-level prediction and as a post-processing tool. The model was evaluated on three public data sets DRIVE, STARE, and CHASE DB1 and the accuracy obtained was 0.9536, 0.9603, and 0.9587 respectively. The sensitivity obtained was 0.7632, 0.7423, and 0.7815 respectively. The model was built and trained on NVIDIA GTX Titan X GPU.

Qummar et al. (2019) [3] proposed an ensemble model which consisted of five deep learning CNN models. The five CNN models in the ensemble are Resnet50, Inceptionv3, Xception, Dense121, and Dense169. The data set mainly used for training was the Kaggle based data set. The pre-processing involved image resizing, data augmentation, and up-sampling and down-sampling. The results obtained on the data set were accuracy, recall, specificity, precision, and F1-score are 0.808, 0.515, 0.8672, 0.6385, and 0.5374 respectively. A method was proposed by Shah et al. (2019) [13] for detecting the blood vessels. In this method, retinal vessel segmentation is done using Gabor filter and multi-line detector. The pre-processing step involves the green channel of the image and then complementing it. Gabor filter is applied to the complemented green channel image. For blood vessel extraction, the multiline detector was used. The model was tested on three public data sets which evaluated the accuracy and sensitivity. The method was evaluated with a publicly available data set called DRIVE, STARE and High-Resolution Fundus (HRF), the accuracy obtained was 0.9470, 0.9472, and 0.9559, and sensitivity was 0.7421, 0.8004, and 0.7207, respectively.

Soomro et al. (2019) [14] devised a model for accurate detection of the retinal blood vessel. For this, the Convolution Neural Network (CNN) was used. The pre-processing methods involved the green channel method for gray-scaling. According to his argument, the green channel possesses the least noise as compared to the blue and red channels. The noise removal was done using the fuzzy C-mean (FCM) method. The pre-processed procedure was completed using CLAHE for a well-contrasted image. FCM model technique was used for the vessel segmentation step. The author used CNN for classification and training of the model using a U-net architecture also known as FCN. The author also went for post-processing to enhance the quality of images. Overall the model was tested on two widely used public data sets DRIVE and STARE. The sensitivity, specificity, and accuracy obtained on DRIVE were 0.802, 0.974, and 0.959 respectively while the same obtained on STARE was 0.801, 0.969, and 0.961 respectively. To overcome the drawbacks mentioned in the existing

researches, the model is proposed to segment the blood vessels by using Gaussian filter as one of the pre-processing steps and for classification VGGnet architecture is used to enhance the results under all the parameters specified.

3. Implementation

The complete architecture of the model is explained in Fig 2. The model goes through different stages. It starts from image preprocessing so that the image does not have any impurities such as noise, uneven contrast illumination, etc. Then the vessel segmentation is done to extract the blood vessels using techniques such as threshold, contouring, erosion, and dilation. Later feature extraction steps are done to refine the vessel segmentation process. And finally, classification and prediction are done using the Convolution Neural Networks model. The color fundus image has been collected from various data sets like DRIVE and STARE.

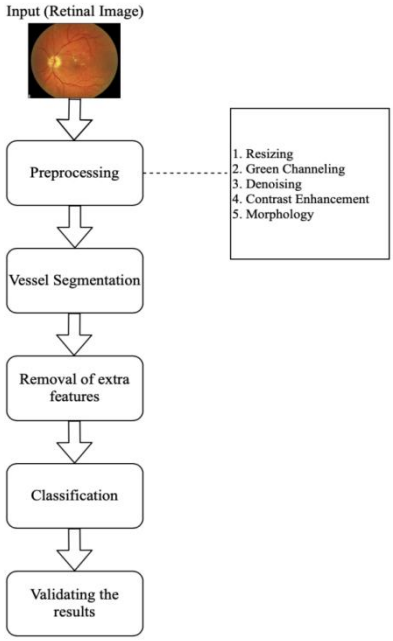

Fig 2. Architecture Diagram

3.1. Pre-Processing

The pre-processing steps includes resizing, green channeling, denoising, contrast enhancement, and morphology. Resizing is the first step of pre-processing and it has been done using an inbuilt python library known as imutils. The image outsize is set with a width of 300. Green channeling is the second step of pre-processing and it converts pre-processed image into mono-color component image. However, the clearest image can be seen in the green channel. As compared to gray scaling, the green channel images give a much clear image. After that denoising is performed using Gaussian filter to remove the random noise and the equation of Gaussian function in 1-D is given below.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2}$$

The above equation represents the Gaussian function in one dimension where x is the distance from the mean and σ is the standard deviation. In a Gaussian blur, the pixel value of each pixel is set to a new value depending on the weighted average of the pixel value of the neighborhood pixel.

Then the contrast enhancement is done to improve the clarity of the image by applying CLAHE. The CLAHE function is implemented as a cv2 function which includes two parameters that are grid size and clip limit. In this proposed work we have set our clip limit to 2 and grid size as 8x8. Finally, the morphology is used and it uses two fundamental operations that are erosion and dilation. In the proposed system, the opening and closing morphology techniques have been used three- times. On each iteration of opening and closing three different parameters have been set. At each iteration, different parameters are passed. In the first iteration, the kernel size of 5x5 was passed. In the second iteration, the kernel size

of 11x11 was passed and in the third iteration, the kernel size of 23x23 was passed. The final morphological results and other pre-processed images are shown in the Fig 3. Fig. 3(a) represents the normal retinal image, 3(b) represents the resized image, 3(c) represents the Gaussian blur image, 3(d) represents the image after applying CLAHE and 3(e) represents the image after applying morphological operations.

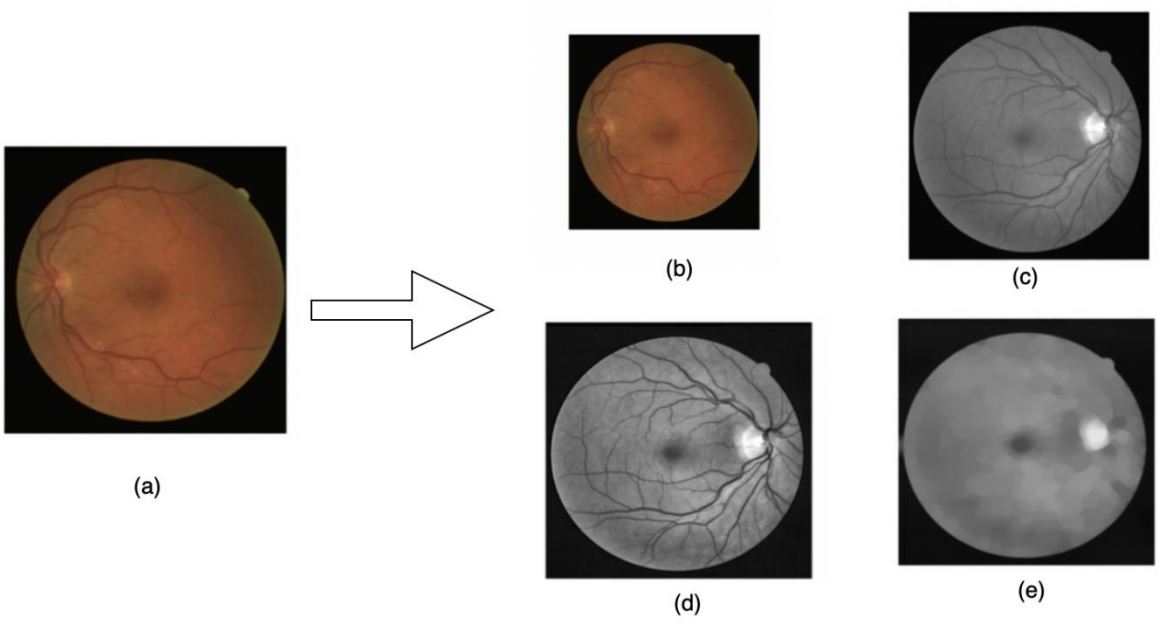

Fig. 3 (a) Normal image, (b) Resized image, (c) Gaussian blur image, (d) Image after applying CLAHE, (e) Morphological opening and closing applied to CLAHE image

3.2 Vessel Segmentation

The vessel segmentation step is the next step after pre-processing. The vessel segmentation is done to extract the blood vessel by extracting the thick and thin lines in the retinal image. In the proposed system thresholding, contouring, erosion, and dilation has been used to segment the blood vessels. In thresholding the value has been set to 15 with the THRESH_BINARY parameter. This helps to distinguish blood vessels from other components since the blood vessel is darker than other components of the retina. Then contouring is done and here three parameters are passed to find contours function. The first argument takes up the source image. The second argument is the mode where RETR_LIST is taken as the model. The RETR_LIST is used for retrieving contours without establishing hierarchical relations. The third argument is the approximation method. For the approximation method, CHAIN_APPROX_SIMPLE is used, which is to remove the redundant points, and hence saves memory and executes faster. Finally, the dilation and erosion are performed and both of them uses the get structuring element function and the ellipse shape is passed as an argument with a kernel size of 3x3 with one iteration.

After vessel segmentation, there may be many chunks and blobs which may be segmented as boundaries or curves. These blobs and chunks are not blood vessels and were segmented as boundaries and curves. In this step, these unwanted features will be removed, leaving only blood vessels and all the resultant images are shown in figure 4. The Fig.4(a), 4(b), and 4(c) represents the pre-processed image, vessel segmented image and image after removing clogs and blobs.

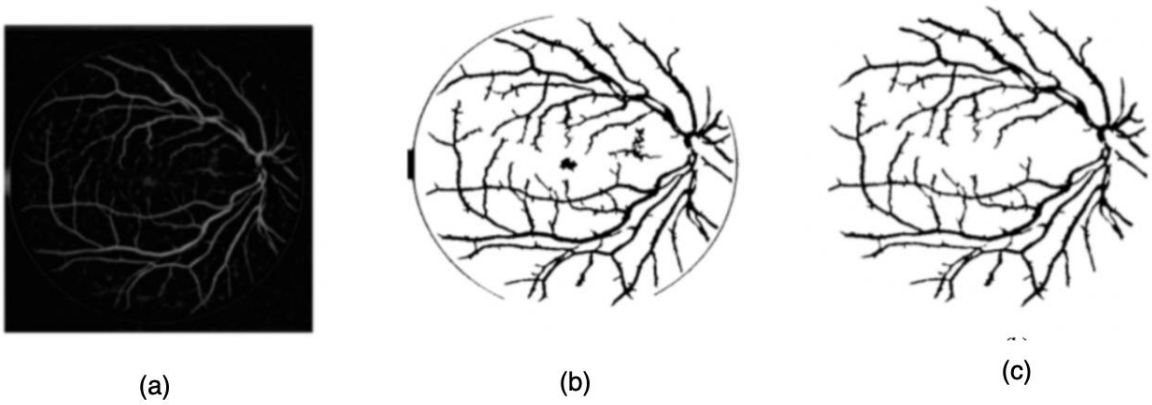

Fig. 4 (a) Preprocessed image, (b) Vessel segmented image, (c) Image after removing extra features

3.3 Classification

The classification of the images is done based on whether the eye is in the non-proliferative stage or proliferative stage. For the classification step we have used neural networks. The neural network model is trained and it learns from the data set given, known as supervised learning. In the proposed work, the CNN or Convolution Neural Network will be used as classifiers. The CNN architecture works in two phases, the first phase is for feature extraction and the second phase is for classification. CNN gives the advantage of combined feature extraction and classification. In the presented model, two famous CNN architectures such as VGGnet architecture [15] and Dense-net 121 have been used. First the classification is performed with VGGnet architecture and then Dense-net architecture is used. The performance of each architecture model is discussed in the results and discussion section. A CNN sequential model has been developed for feature extraction and classification. In the sequential model, the different layer has aligned one after the other. Fig. 5 shows the sequence layer structure of the architecture.

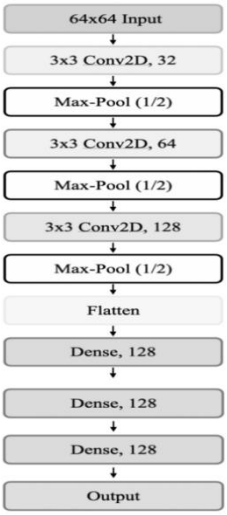

Fig 5. Sequential model layers

There are four layers used in the model and the kernel size is taken to be 3x3. In each layer of Convolution 2D, the size of the kernel is maintained to 3x3. The model is initiated with a sequential layer and an input image size of 64x64. The first convolution 2D layer has a depth of 32. Each depth also known as the number of filters will generate a feature map. Feature maps are generated by the result of the dot product of the kernel with each filter. Since there are 32 filters, 32 feature maps will be generated. The activation of the 'ReLu' or Rectified Linear unit has been defined in the Convolution 2D layer. The next layer is the Max-pool 2D layer. The max-pool takes up pool size as 2x2. This value represents that the image with kernel size 2x2 will be max-pool. The next layer is a Convolution 2D layer with a depth of 64. In a general VGGnet model, the depth should increase by two times with every convolution layer. Hence the depth is taken to be 64. This layer has also been supported by a 'real' activation. This layer generates 64 feature maps. The next layer is the Max-pool 2D layer with the same

pool size of 2x2 and performs the same function. The next layer is a Convolution 2D layer with a depth of 128 with 'ReLu' activation. The next layer is the Max-pool 2D layer with the same pool size of 2x2. The combination of Convolution 2D layers and Max-pool 2D layers form the feature extraction part of the CNN. The feature maps generated throughout the process is what detects the extracts of the different features from the data set images.

The next part of CNN is a classification that is supported by Flatten and Dense layer. All the data that has been acquired from the feature extraction part are flattened into a single dimension layer. For this the Flatten layer has been used. The next layer used is the Dense layer. The dense layer is a fully-connected layer with 'ReLu' activation. In the dense layer, each input neuron is connected to the output neuron. The size of the layer has been set to 128. There are 3 dense layers used with activation 'ReLu' and the size of the layer as 128. Finally, the output dense layer is set with 'sigmoid' activation. A VGGnet architecture is simple and takes less computation time. The architecture is still considered to be one of the best when it comes to training as it takes less computation time and power and it is best suited for binary classification. The VGGnet architecture is shown in the Fig.6.

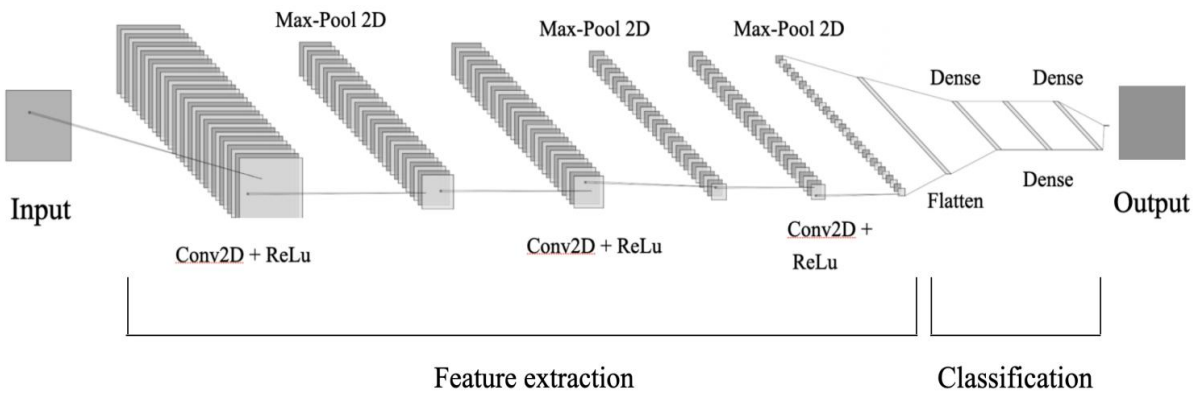

Fig 6. VGG net architecture

Another architecture used in the proposed model is Dense-net 121 architecture. Dense-net 121 is a CNN architecture that is very effective in categorical classification and training. Dense-net 121 is composed of Dense blocks. The architecture works in a feed-forward fashion. The blocks in architecture are densely connected, hence the name Dense-net. Dense-net 121 has 121 layers. The feature maps of the previous layer are added to the next layer. The architecture has three major blocks. The major function of a dense block is to concatenate the inputs received. The convolution block has three layers that are batch normalization, 'ReLu' activation and convolution 2D layers. The transition layer works as a transition between different dense blocks by down sampling using a convolution layer along with an average pool layer. Due to the densely connected blocks, the training and test loss is very less.

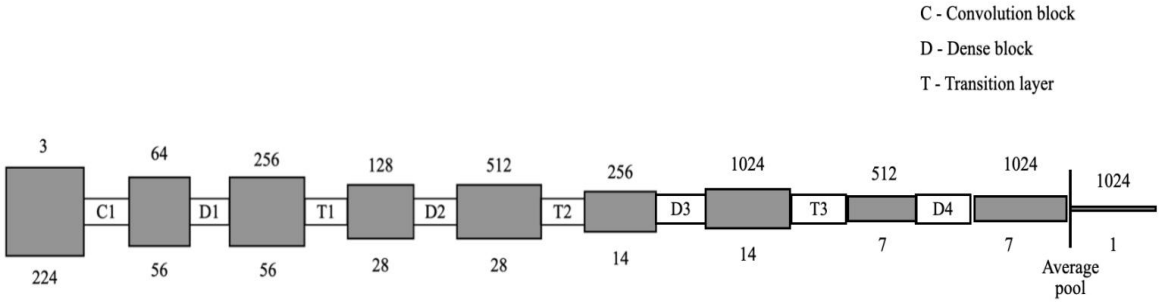

Fig 7. Flow of a Dense net architecture

Fig. 7 shows a structure of a connection between dense net blocks. Though the Dense-net architecture has several advantages the significant improvements could not be obtained over the existing systems in the proposed work. The performance of the Dense-net architecture is very low compared with VGGnet architecture. The time taken by this model for the execution is 86537.26310515404 s which is much greater than the VGGnet architecture model.

3.4 Prediction

In the proposed system, the model is used to determine whether the retinal image is affected with neovascularization, that is it belongs to the proliferative stage or not. To implement this model, the prediction module is developed. The prediction module generates a GUI in which an image is uploaded and the GUI reflects the result whether the retinal image is in a proliferative stage or in the non-proliferative stage. The prediction model determines the results in terms of accuracy, recall, precision, specificity, sensitivity, and F1-score. These evaluation metrics are determined on two data sets that are DRIVE [9] and STARE [10]. The actual value of the images is fed in an array and then the prediction module runs to give an array of predicted stages and accordingly gives the parameter. The prediction module basically predicts the stage of an image based on the classification and training done. The training model is saved in the HDF5 files used in python [16]. This HDF5 file is then used by the prediction module. The use of an HDF5 file for saving the training model helps to avoid repeatedly training the same model again and again and also highly reduces the computational time. The module gives a confusion matrix determining the true positive (TP), true negative (TN), false positive (FP), and false-negative (FN). Based on these four results obtained due to the confusion matrix, the evaluation parametric is calculated.

4. Results and Discussions

The model execution was performed on Anaconda Spyder version of 3.3.6 and python version 3.7. The model execution also required Keras version 2.3 and TensorFlow version 1.14. The system is configured with Intel(R) Core (TM) i7-6700HQ Cpu@2.60 gHz, 16 GB RAM, Nvidia GeForce GTX 960 GPU.

4.1 Datasets used

The Kaggle data set has been used for training. A training set of 1600 images with 800 images each of the two classes. The two classes are defined as NPDR and PDR. The test set consists of 319 images with 200 images in each of the two classes. The model was tested on two most popular public data sets for blood vessel segmentation that is DRIVE and STARE. The DRIVE consists of 40 images, 20 for training and 20 for testing [9] while the STARE consists of 20 images all together [10].

4.2 Evaluation Metrics

The proposed model was tested on the two most popular public data sets that are DRIVE and STARE. The performance on both the data set was measured using six factors that are accuracy, precision, recall, sensitivity, specificity, and f-score. The result obtained on these data sets is explained in table 1. The factors can be calculated based on the confusion matrix. The below data gives the formula to calculate the performance of each of the factors where TP is true positive, TN is true Negative, FP is false positive and FN is false negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1 - Score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

4.3 Performance evaluation with VGGnet architecture

The Keras sequential model is used with the Keras version of 2.3. The training and testing of the images are done using two sets of data that is the training set and testing set. The training set consists of 1600 images belonging to 2 classes and the testing set consists of 319 images belonging to 2 classes altogether. The steps are the number of steps it takes to complete one epoch. The steps are set to 500. One epoch is completed when the entire data set is sent for computation. Here, the number of

epochs is set to 25. The maximum accuracy reached using DRIVE and STARE images are 96% and 95% respectively. The others factors such as precision, sensitivity, specificity, and F1-score are shown in Table 1.

Table 1. Performance evaluation on public data sets using VGGnet architecture

	Accuracy	**Precision**	**Sensitivity**	**Specificity**	**F1-score**
DRIVE	0.96	0.99	0.95	0.99	0.97
STARE	0.95	0.96	0.9375	0.99	0.95

4.4 Training and Testing results of VGGnet and Dense-net 121 architecture

The comparison between training accuracy vs epoch and test accuracy vs epoch using VGGnet architecture is shown below.

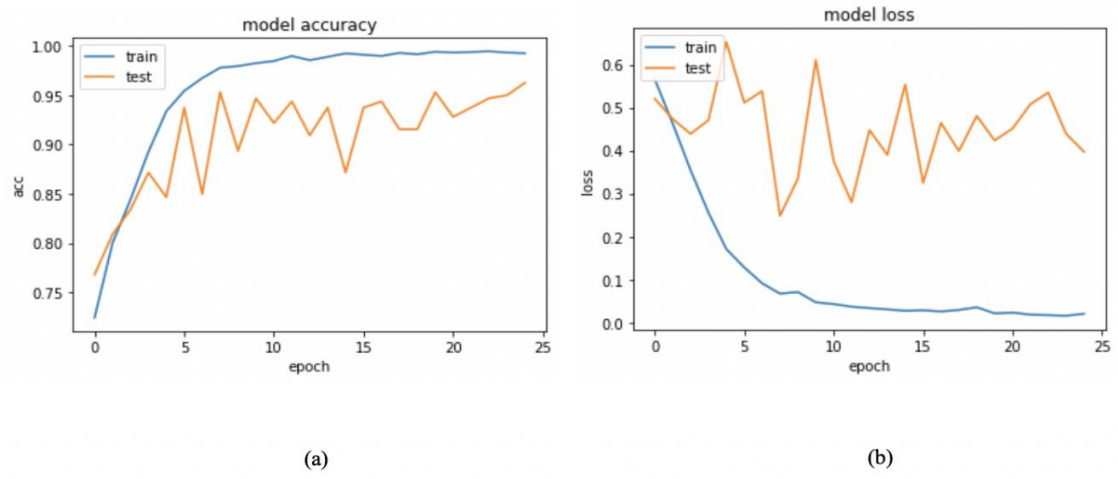

Fig 8. (a) Training accuracy v/s Test accuracy of VGGnet architecture, (b) Training loss v/s Test loss of VGGnet architecture

Fig 8(a) graph shows a comparison between the training accuracy and test accuracy. As shown in the graph the training goes through 25 epochs. Each epoch has been trained in a batch size of 32. The training accuracy achieved is 0.998 while the test accuracy achieved is relatively less around 0.9624. This accuracy achieved on the training data set is relatively high in relation to the data taken for training and classification. The approximate range of the validation accuracy and training is quite high and the model has high evaluation metrics due to the results obtained from training and validation accuracy. Fig 8(b) graph shows a comparison between the training validation loss and test validation loss. As shown in the graph the training goes through 25 epochs. The training validation loss achieved is 0.0186 while the test validation loss is 0.3846. The validation loss achieved on both training and test is relatively good. The loss on the training set is very less while it is more on the test set since the data is less on the test set as compared to the training data set as mentioned in the data acquisition.

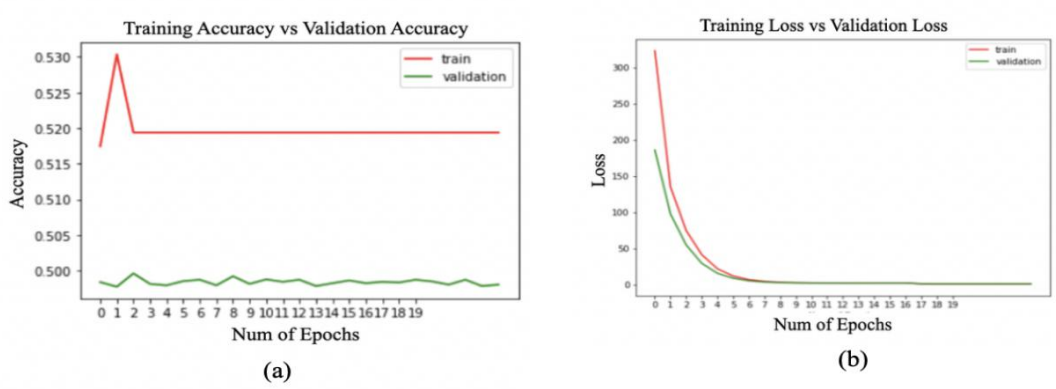

Fig 9. (a) Training accuracy v/s Validation accuracy for Dense-net 121 architecture, (b) Training loss v/s Validation loss for Dense-net 121 architecture

Fig 9(a) gives the graph for training accuracy v/s validation accuracy and Fig 9(b) gives the graph for training loss v/s validation loss for the Dense-net 121 architecture. As shown in the graph the training goes till 25 epochs and the training and validation accuracy achieved using the Dense-net model is 0.5194 and 0.4980 respectively which is very less compared with VGGnet architecture.

4.5 Comparison of VGGnet and Dense-net 121 model

In this section, the comparison between the two most famous CNN architecture VGGnet and Dense-net 121 is done. There are different parameters on the basis of which the two models are compared. Both the models are trained over 2200 images all together and train for 25 epochs for uniformity. Training for a lesser number of epochs solves the problem of overfitting.

Table 2: Comparative analysis of VGGnet architecture and Dense-net 121 architecture

Parameters	VGGnet	Dense-net 121
Training accuracy	0.9982	0.5194
Validation accuracy	0.9624	0.4980
Training loss	0.0186	0.8481
Validation loss	0.3846	0.8593
Computational time	1601.7414999008179 s	86537.26310515404 s

Table 2 gives a comparison of the two architectures based on different parameters. The inference generated from the stimulation of the two architectures is given as follows. The Dense-net 121 architecture clearly gives a much less training accuracy that is 0.5194 and validation accuracy that is 0.4980 as compared to VGGnet where training accuracy is 0.9982 and validation accuracy is 0.9624. Validation accuracy and loss are also known as test accuracy and loss. The training loss that is 0.8481 and validation loss that is 0.8593 generated is also much higher compared to training loss that is 0.0186 and validation loss that is 0.3846. The computational time taken by Dense-net 121 is relatively much greater that is 86537.26310515404 s (24 hours) more almost fifty-five times the computational time for VGGnet that is 1601.7414999008179 s (26.6 min appx).

The major drawback derived from the stimulation of Dense-net 121 architecture was that the predictions for the NPDR class are quite accurate but the predictions for PDR class are absolutely incorrect. The Dense net 121 model predicts all PDR class images as NPDR which comes as a major drawback for the research being done. The VGGnet model on the other hand gives quite relatively accurate predictions for both the classes. The computational training time being taken to almost 1 day is also a limitation. The much lower training and validation accuracy and higher training and test loss come as a major drawback. In Dense-net 121 the learning rate also drops highly to about 0.00010000000474974513 after the 17 epochs.

After the 17 epochs, the graph becomes constant, hence the epoch range in the graph is shown till 19 epochs only, though the training is done till 25 as shown in Fig 20 and Fig. 21. Both Dense-net 121 and VGGnet are famously used in the architecture of CNN but for this research the VGGnet stands out better than Dense net-121. Hence this is the comparative analysis for Dense-net 121 architecture and VGGnet architecture. The Dense-net 121 works better for complex structures with many features and a greater number of classes.

4.6 Comparison of classification results with existing methods

Table 3 and Table 4 demonstrates the performance measures of different works done by different researchers using DRIVE and STARE data set. It clearly shows that our proposed model achieved better results than all other models.

Table 3. Performance measures of the proposed model on DRIVE data set with existing methods

DRIVE DATASET					
	Accuracy	**Specificity**	**Sensitivity**	**Precision**	**F1-score**
S.S. Kar and S.P. Maity (2018) [4]	0.95	-	-	-	-
S. Thangaraj et al. (2018) [11]	0.96	0.97	0.80	-	-
Yi Lin et al. (2019) [12]	0.95	-	0.76	-	-
Shah et al. (2019) [13]	0.94	-	0.74	-	-
T. A. Soomro *et al*. (2019) [14]	0.95	-	0.80	-	-
Xiuqin Pan et al. (2019) [16]	0.96	0.98	0.93	-	-
Zengqiang Yan et al. (2019) [17]	0.95	0.98	0.76	-	-
D.A. Dharmawan et al. (2019) [18]	-	0.97	0.83	-	0.82
V.Cherukuri et al. (2020) [19]	0.95	0.98	0.84	-	0.82
Proposed Model	**0.96**	**0.99**	**0.95**	**0.99**	**0.97**

Table 4. Performance measures of the proposed model on STARE data set with existing methods

STARE DATASET					
	Accuracy	**Specificity**	**Sensitivity**	**Precision**	**F1-score**
S.S. Kar and S.P. Maity (2018) [4]	0.95	-	-	-	-
S. Thangaraj et al. (2018) [11]	0.94	0.95	0.83	-	-
Yi Lin et al. (2019) [12]	0.96	-	0.74	-	-
Shah et al. (2019) [13]	0.94	-	0.80	-	-
T. A. Soomro *et al*. (2019) [14]	0.96	-	0.80	-	-
Zengqiang Yan et al. (2019) [17]	0.96	0.98	0.77	-	-
D.A. Dharmawan et al. (2019) [18]	-	0.98	0.79	-	0.81
V.Cherukuri et al. (2020) [19]	0.96	0.98	0.86	-	0.83
Proposed Model	**0.95**	**0.99**	**0.9375**	**0.96**	**0.95**

The tables above depict that most of the paper has concentrated on accuracy, specificity, and sensitivity. In the future, a greater number of data sets will be tried on to get better performance metrics. Hence, the overall results achieved with this model have been quite satisfactory. The evaluation metrics have been better as compared to previous research done. VGGnet is a simple architecture with a sequential model. The computation time achieved is far less than the Dense net architecture. VGGnet also gives relatively good results on all evaluation parameters. The Gaussian filter proves to be better for noise removal as compared to the Gabor filter. The training and testing accuracy and loss values are also quite satisfactory. Both DRIVE and STARE have shown good evaluation metrics value. The work is supported by relevant graphs to give more clarity. Overall, the research has shown a good result altogether.

5. Conclusion

The researches being made on blood vessel segmentation and neovascularization are relatively less as compared to other features of NPDR. Different pre-processing techniques help to get better results. The comparison between different CNN models shows that different types of architecture suit different projects, based on number of images and number of classes. Hence no architecture is best or perfect. The main challenges that occur in these areas of research are that the data set available for neovascularization or proliferative stage is relatively very less. Hence the proper training and classification are subjected to the limited availability of data sets. Another limitation arrives with the time taking processor that takes way more time to compute the process. The detection of the tiny blood vessel is another challenge which reduces performance results on the images. The proposed system overall focuses on all the evaluation metrics with a high score on each. The data set used is a public data set that gives the audience an open end for research and comparison. In the future, more work is expected in an increase in performance and further approach to work on a greater number of data sets is expected.