Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Research of Retinal Vessels Segmentation by Fully Convolutional Networks

Master's Final Degree Project

Divyesh Mune Gowda

Project author

Assoc.Prof. Arunas Lipnickas

Supervisor

Kaunas, 2021

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Research of Retinal Vessels Segmentation by Fully Convolutional Networks

Master's Final Degree Project

Control Technologies (6211EX014)

Divyesh Mune Gowda

Project author

Assoc.Prof. Arunas Lipnickas

Supervisor

Prof. Vidas Raudonis

Reviewer

Kaunas, 2021

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Divyesh Mune Gowda

Research of Retinal Vessels Segmentation by Fully Convolutional Networks

Declaration of Academic Integrity

Divyesh Mune Gowda

Confirmed electronically

Author’s Mune Gowda, Divyesh. Research of Retinal Vessel Segmentation by Fully Convolutional Networks. Master’s Final Degree Project supervisor Assoc.Prof. Arunas Lipnickas; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): Electronics Engineering (engineering sciences)

Keywords: Convolution Neural Network, UNET, Retinal vessel segmentation.

Kaunas, 2021. 65 pages.

Summary

In this research, the retinal vessel segmentation by a fully convolutional neural network is studied in detail. Retinal vessel segmentation is helpful in the diagnosis of diabetic retinopathy, hypertension and arteriosclerosis. This research aims to improve convolutional neural networks to provide efficient retinal vessel segmentation results. The UNET was introduced in 2015, which provided better efficient segmentation for biomedical images with a fewer dataset. In this research, the UNET and possible modified and improved networks based on UNET was built and tested for efficient segmentation of retinal vessels. A fully convolution network UNET G UCDA is proposed in this project which was found during the research to provide efficient retinal vessel segmentation compared to all other UNET based networks that were trained and evaluated in this research.

Autoriaus Mune Gowda, Divyesh. Tinklainės kraujagyslių segmentavimo tyrimas taikant konvoliucinius neuroninius tinklus. Magistro baigiamasis projektas vadovas Doc. Arunas Lipnickas; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektronikos inžinerija (inžinerijos mokslai).

Reikšminiai žodžiai: Convolution Neural Network, UNET, tinklainės kraujagyslių segmentavimas.

Kauno, 2021. 65 p.

Santrauka

Šiame tyrime išsamiai tiriamas akies tinklainės kraujagyslių segmentavimas taikant konvolucinį neuroninį tinklą. Tinklainės kraujagyslių segmentavimas yra naudingas diagnozuojant diabetinę retinopatiją, hipertenziją ir arteriosklerozę. Šis tyrimas rodo kaip pagerinti konvoliucinius neuroninius tinklus, kad būtų gauti kuo geresni tinklainės kraujagyslių segmentavimo rezultatai. UNET buvo pristatytas 2015 m., Kuris užtikrino efektyvesnį biomedicinos vaizdų segmentavimą apmokinus jį su mažu duomenų rinkiniu. Atliekant šį tyrimą, UNET ir galimi modifikuoti bei patobulinti tinklai, pagrįsti UNET, buvo sukurti ir išbandyti efektyviam tinklainės kraujagyslių segmentavimui. Šiame projekte siūlomas visiškai konvoliucinis tinklas UNET G UCDA, kuris buvo nustatytas tyrimo metu, siekiant užtikrinti efektyvų tinklainės kraujagyslių segmentavimą, palyginti su visais kitais UNET pagrįstais tinklais, kurie buvo apmokyti ir įvertinti šiame tyrime.

Table of contents

List of figures

List of tables

List of abbreviations and terms

Abbreviations:

ROI	– Region Of Intrest.
RBVS	– Retinal Blood Vessel Segmentation.
ANN	– Artificial Neural Networks.
CNN	– Convolutional Neural Networks.
DCNN	– Deep Convolutional Neural Networks.
FCN	– Fully Convolutional Neural Networks.
UNET	– U shaped Convolutional Neural Network.

Introduction

The task of image segmentation consists of partitioning an input picture into non-overlapping regions. It aims to highlight some regions of interest (ROIs) for a more in-depth study/analysis on medical images. The segmentation result can then be used to extract various morphological parameters of the ROIs, which can help with disease diagnosis and control. The blood vessels and the optic disc are included in the ROIs on a retinal image, as shown in Figure 1. Diabetic retinopathy, hypertension, and arteriosclerosis can all be diagnosed, treated, and monitored using retinal blood vessel segmentation (RBVS).[1][2]. Optic disc segmentation (ODS) can bring relevant insights into glaucoma disease [2]. In present days, these diseases are the leading factors of blindness [3]. In this research, we will be mainly concentrating on RBVS.

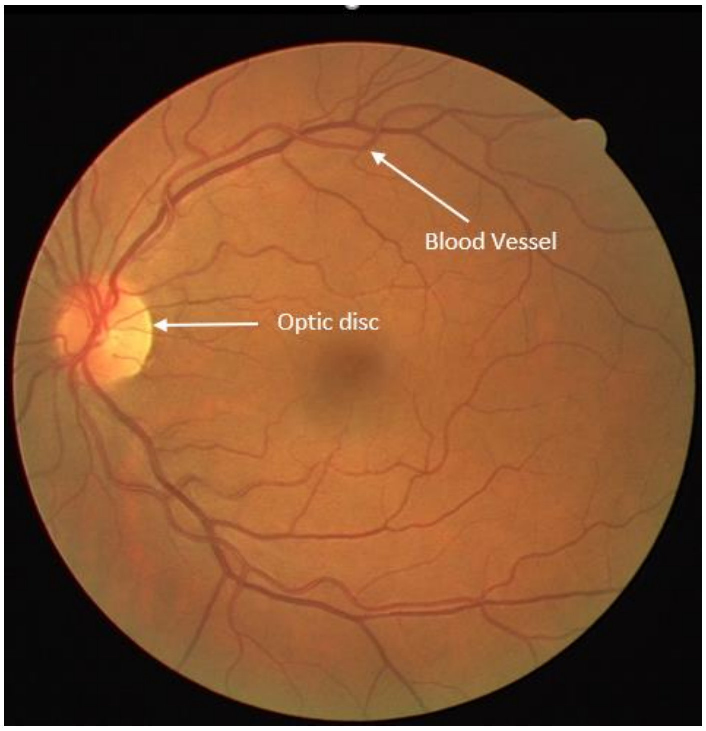


Fig. 1. Retinal Image

Annotation of the blood vessels manually is a time-consuming and challenging task that necessitates expertise. Expert segmentations are often subject to inter-and intra-operator variability, making them inefficient for large-scale and real-time applications. As a result, a significant amount of effort has gone into developing automated processes. [4][5].

For retinal image segmentation, we will be using UNET based architecture. UNET architecture is developed to yield more precise segmentation with fewer training images.

UNET, which developed from the conventional convolutional neural network, was designed and used to process biomedical images for the first time in 2015 [16]. A general convolutional neural network focuses on image recognition, with an image as input and a single label as output. In biomedical cases, however, we must not only determine if there is a disease but also pinpoint the location of the abnormality. UNET is committed to resolving this problem. It can localise and distinguish borders by classifying every pixel, so the input and output share the same size. The network is based on a

FCN , and the network architecture was modified and extended to work with fewer training images/data and yield more precise segmentation.

The aim of this project is to improve and modify UNET for efficient segmentation of Retinal Blood Vessels Segmentation (RBVS).

Steps followed to achieve the aim:

1. Research on CNN‘s.
2. Study of UNET structure.
3. Study of UNET with basic modifications like depth and filter sizes.
4. Modification and improvisation of the best performing UNET model. Here the modifications were done based on CNN structural research and other possible modifications suggested by other research papers.

1. Background :

There have been many research activities in implementing efficient retinal image segmentation. From the retinal image, diabetic retinopathy can be easily diagnosed. Diabetic retinopathy is widely researched and implemented. Many algorithms based on DCNN and FCNN are used in retinal image segmentation and diagnosis of diabetic retinopathy. Since the data the models are trained on is significantly less, these results are not efficient. There is also more research in retinal image segmentation to diagnose hypertension, arteriosclerosis and other diseases. Medical images cannot be easily trained on DCNN and FCNN since every image is unique. The neural network models have to be modified for specific cases in order to get good results. Because of these problems, it has not yet been implemented for real-time and large scale uses. There is more improvement and modification necessary for implementing neural networks for real-time and large scale uses.

1.1. Retinal imaging and Analysis :

The retina tissue forms the inside of the eye. It helps to convert the light reflected on it into a neural signal that is then sent to the brains visual cortex to be further processed. As a result, it acts as a brain extension. Researchers are intrigued by the ability to photograph the retina and develop techniques for analysing the pictures. Since the retina in the human body functions as a tool to see the outside world, the methods involved in the image creation of the retinal must be optically transparent. As a result, the retina can be seen from the outside with appropriate techniques, allowing noninvasive visualisation of retinal tissue and thus brain tissue (Figure. 2). The retina also has a double blood supply that allows for noninvasive bloodstream monitoring [1].

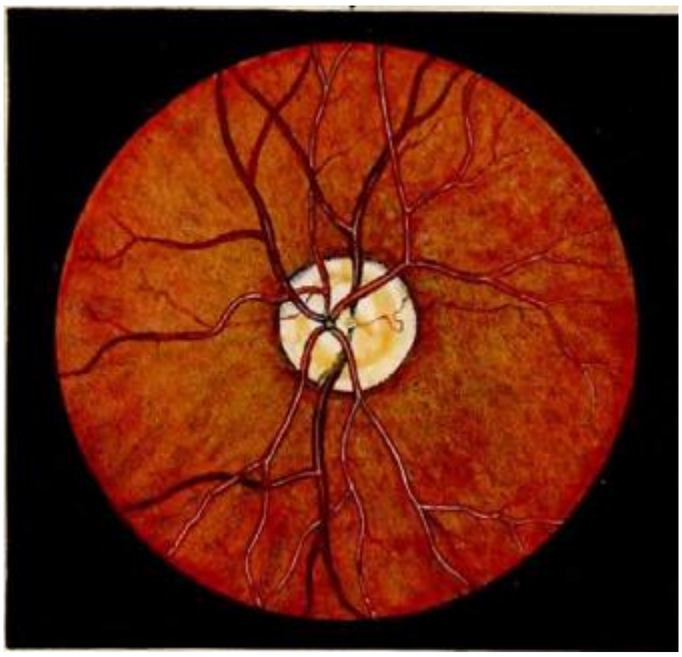


Fig. 2. First image of the human retina[1]

With the invention of the Ophthalmoscope in the 1850s, the evaluation of the retinal structure by Ophthalmologist became a familiar and routine task. The first images of the retina (Figure. 2) was published in 1853 by van Trigt [1].

Over the last 160 years, retinal imaging has progressed slowly, and it is now an important part of the treatment and management of patients with retinal and systemic diseases. In vast populations, fundus imaging is widely used to diagnose diabetic retinopathy, glaucoma, and age-related macular degeneration [1].

Glaucoma and Diabetic retinopathy are among the leading causes of blindness in the modern world. The number of people affected by these diseases is gradually increasing. Hence the demand for experts is also is increasing for proper diagnoses.

1.2. Retinal Image Segmentation :

Diabetic retinopathy, hypertension, and arteriosclerosis can all be diagnosed, treated, and monitored using retinal blood vessel segmentation (RBVS). [2][3] To achieve the task of segmentation, researchers and scientist have developed numerous methods over the years.

The RBVS of the eye has been a trending research topic in recent years. The research field aims to develop automated computer-aided technologies to extract retinal blood vessels for the screening and diagnosis of diseases. Despite the development of many promising techniques and algorithms, blood vessel segmentation methodologies can still be improved [4]. Few of the computer-aided retinal segmentation methods are proposed, like the use of texture parameters and gaussian mixture model with support vector machine (SVM) by authors of [6] [13] and multilayer thresholding by authors of the paper [14].

Despite the fact that several papers have been published in recent years on automated RBVS, there is still more improvement needed. The majority of previous approaches dealt with fewer, often healthier images, usually segmenting the larger retinal blood vessel and much lower quality input images. Segmentation in the presence of anomalies, segmentation in non-uniform illumination, and correct segmentation of thin retinal blood vessel are some of the problems that remain open to the research community [5].

1.3. Artificial Neural Network (ANN):

ANN‘s are inspired by biological neurons or neural networks. The mathematical model that was built based on biological neural networks is called ANN. A neural network is a set of artificial neurons that work together to process data using a autoassociative approach to computation.

ANN‘s are massively parallel computing systems. They are made up of very large number of simple processors connected by a large number of interconnections [24]. Each neuron in the network has the ability to receive, process, and transmit input signals. ANN models try to mimic some of the organisational principles found in humans. A theoretical model based on natural neurons is known as an artificial neuron. A simple example of ANN is a simple Feed Forward Network as shown in figure 3 [25].

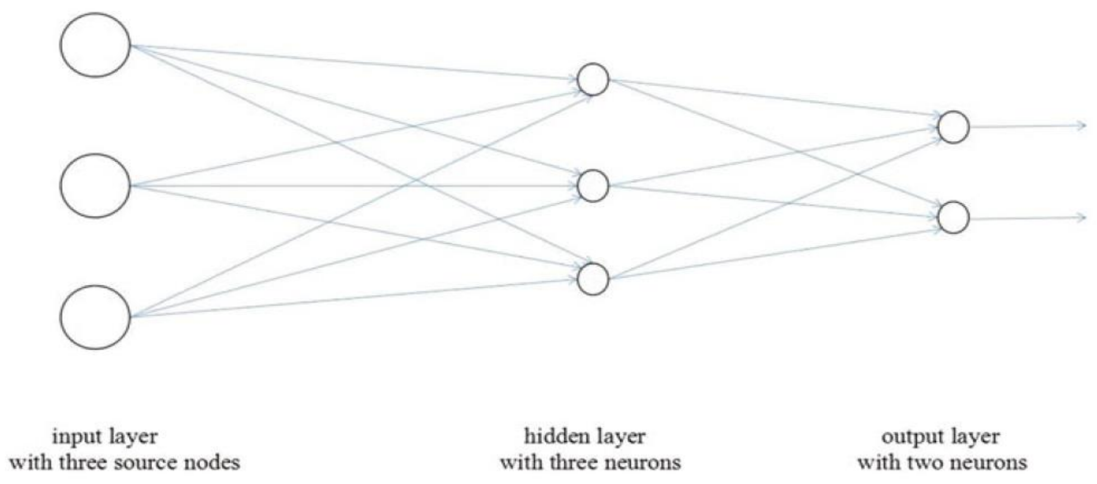


Fig. 3. Simple Feed Forward Network[24].

An input layer of source nodes, one or more hidden layers, and a layer of neurons comprise the feedforward network. The network’s hidden layers cannot be seen directly from the network’s input or output layers. The neural network uses these hidden layers to derive higher-order statistical features from its data. Hidden neurons are neurons that are found in hidden layers. These hidden neurons have the ability to intervene in any way between external input and network output[24].

ANNs are used to model real neural networks and to study animal and computer behavior and control. Pattern detection, forecasting, and data compression are among the computing applications for which they are used. The ANN‘s have become popular in recent years in research and real world problem solving applications. ANN‘s posses capabilities that surpasses other technologies in similar applications.

1.4. Convolutional Neural Network (CNN):

The CNN‘s are a form of ANN that is frequently used to analyse images. In the same way as traditional ANN‘s are made up of neurons, CNN‘s are made up of neurons that learn to optimise themselves. The basis of countless artificial neural networks is that each neuron will always perform an activity after it receives feedback. From the input to the final output, the entire CNN will express a single observant score function (weight). The loss functions associated with the groups will be included in the final layer, and all of the usual ANN operations will still apply [26].

The difference between CNN and ANN is that CNN‘s are mostly used in image recognition, allowing image-specific features to be encoded into the network, making it ideally suited for tasks focused on image.

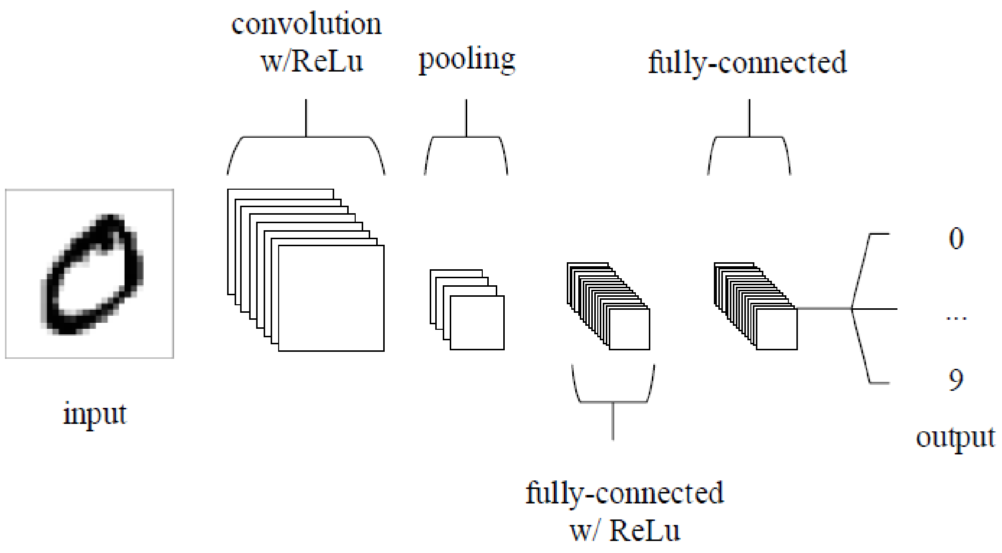


Fig. 4. Simple Convolutional Neural Network [26]

Figure 4 shows a simplified CNN architecture. The convolutional layer consists of four essential parts: Input layer, Convolutional layer, Pooling layer and Fully connected layer [26].

The input layer stores the pixel values of the images. The convolutional layer determines the output of neurons connected to input. The pooling layer will simply downsample the data, taking into account its spatial dimensionality. The completely linked layers will then attempt to generate class scores from the activations that will be used for classification.

The convolution networks are used in image segmentation applications like document recognition [9] and biomedical image segmentations like retinal vessel segmentation, cell segmentation, and many others.

1.5. Dens Convolution Network :

When CNN’s become more complex and have more layers, a new issue arises: when data about the input moves through several layers of the network, it can be lost by the time it reaches the output of the network. To solve this problem and ensure maximum information flows through all the layers of the network authors of the paper [19] proposed a Dense Network.

To ensure efficient knowledge flow, all layers in the network are connected directly to one another. To retain the feedforward nature of the network, each layer contains additional inputs from all the other preceding layers and passes on its own feature to all the other subsequent layers. The schematic representation is shown in figure 5. In this scheme, the features are concatenated in order to combine them. As a consequence, the l^{th} layer in the network has l inputs, each of which is a feature from the preceding convolutional blocks. Both L-l subsequent layers receive their own feature maps. L(L+1)/2 connections are introduced in an L-layer network, instead of just L, as in typical architectures. This dense network pattern is referred to as Dense Convolutional Network.

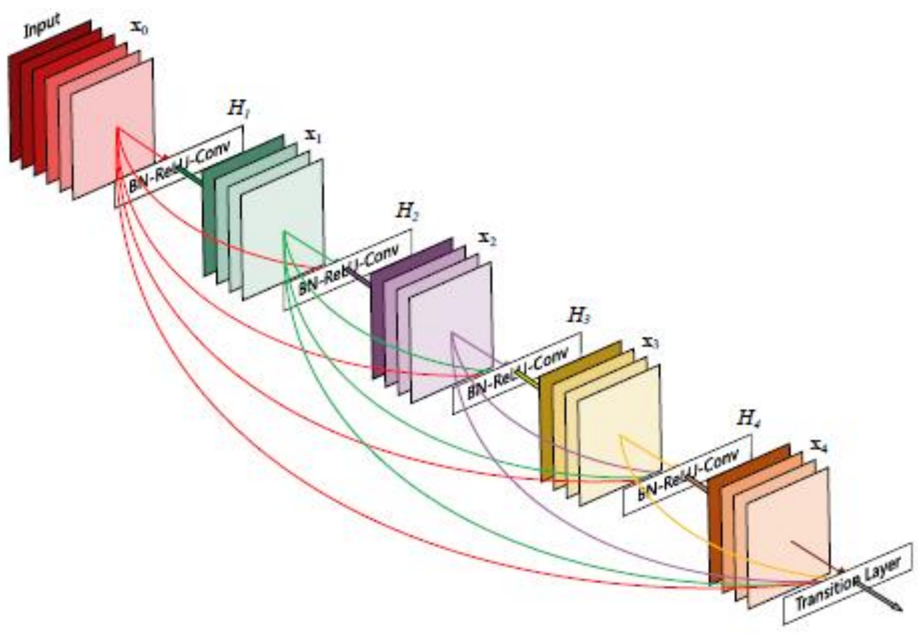


Fig. 5. 5 layer Dens Block [19].

1.6. Atrous Convolution :

The use of CNN‘s for image segmentation and other prediction tasks has been demonstrated to be simple and effective when CNNs are deployed. The combination of maxpooling and striding operations at successive layers of CNNs greatly reduces the spatial resolution of the features. The use of ‘deconvolutional’ layers is a partial solution, but it takes more memory and time. This problem is solved by using atrous convolution. This technique was initially developed for the efficient wavelet transform computation [22]. It was used in the CNN by authors of [30], [31]. This algorithm allows for the computation of any layer’s responses at any resolution. It can be used while training or after training the network. Figure 6 shows feature extraction on a high-resolution feature-map using one-dimensional Atrous convolution with rate of two.[23].

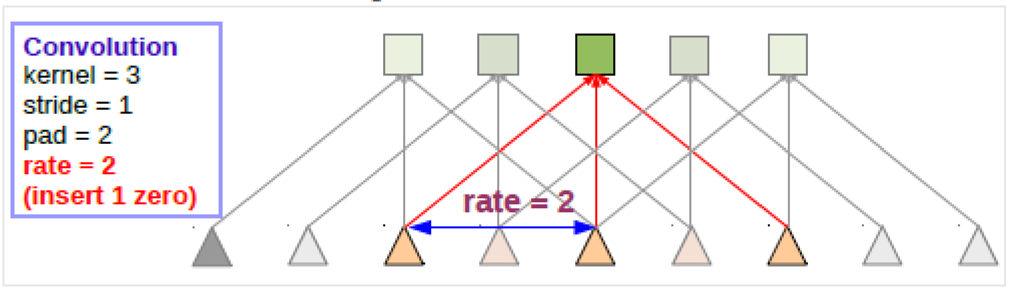


Fig. 6. 1D Atrous Convolution [23].

A variant of atrous convolution is proposed by authors of paper [23], inspired by spatial pyramid pooling by paper [32]. Atrous Spatial Pyramid Pooling (ASPP) is the result of combination of Atrous Convolution and spatial pyramid pooling. Figure 7 shows how ASPP uses multiple parallel filters with different rates to identify the center pixel, which is shown in orange color. The Field-of-views are shown in various colours. Atrous spatial pooling was also used for other images segmentation applications by authors of the paper [20] and [21].

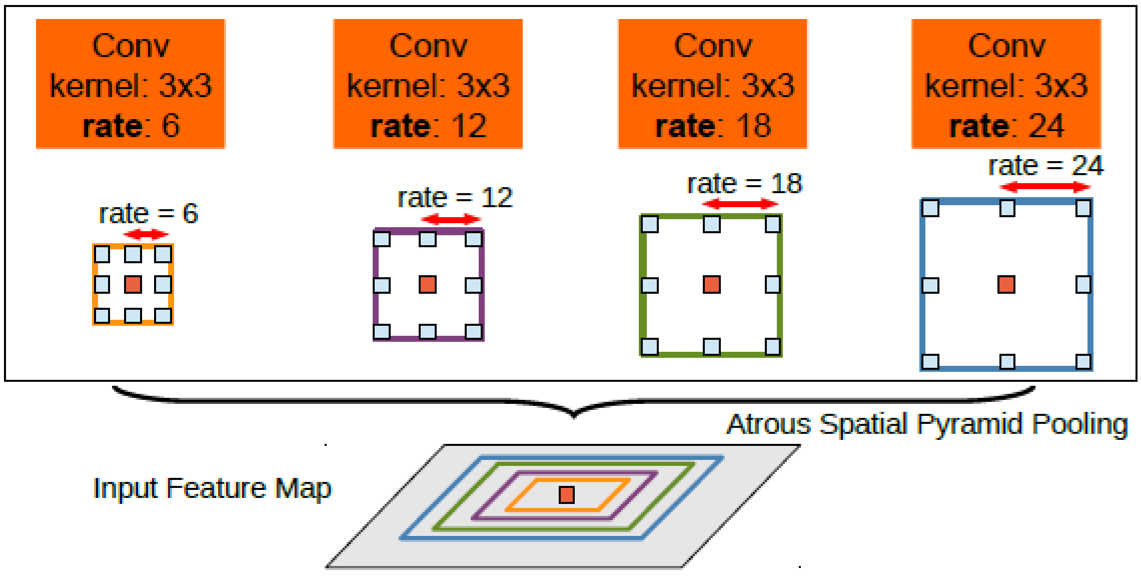


Fig. 7. Atrous Spatial Pyramid pooling [23].

1.7. Convolutional Neural Networks in Retinal Segmentation :

With the introduction of Machine learning, there has been significant improvement in retinal vessel segmentation. Various CNN and FCN networks have been applied to generate retinal segmentation.

A multiscale FCN model with Stationary Wavelet Transform (SWT) was proposed by authors of the paper [18]. The proposed method consists of four stages: input building by SWT, retinal patch extraction, FCN classification, and predictions. The SWT was used to enrich the input to the FCN. The proposed FCN model is shown in figure 8. For training the FCN model, 2750 patches were used from each image of DRIVE. The output retinal vessel segmentation is shown in figure 9. The model evaluation on test images produced an accuracy of 0.9576.

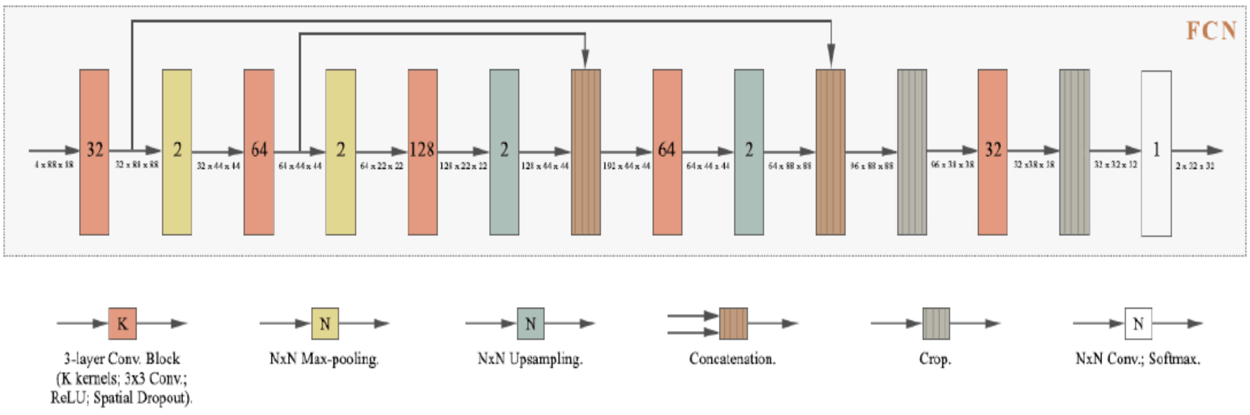


Fig. 8. FCN Model [18]

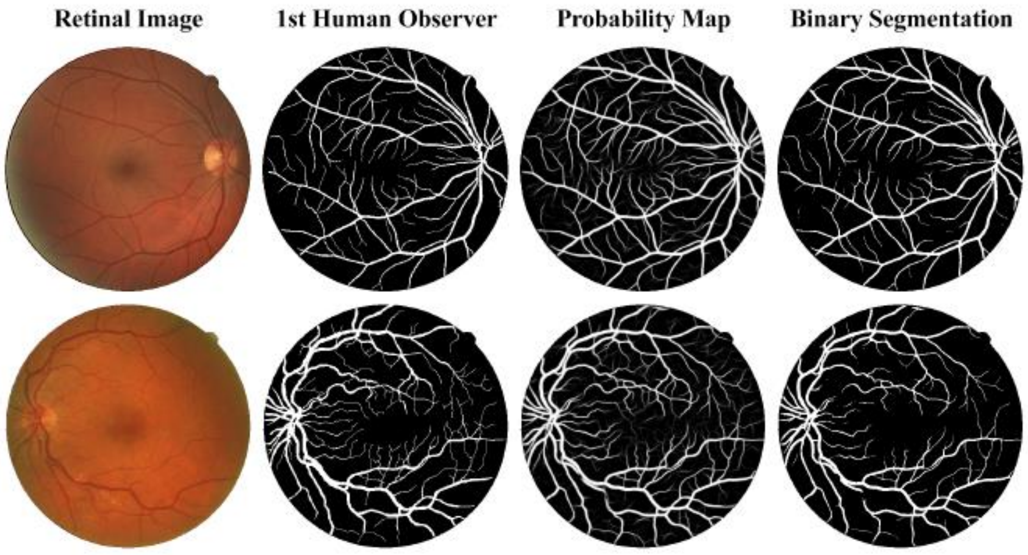


Fig. 9. Output Retinal Segmentations of SWT with FCN [18]

A few other FCN models used for obtaining better retinal vessel segmentation are also proposed by authors of papers [29] and [33], suggesting that data augmentation and introduction of skip connections in FCN can produce better segmentation results. Implementation of ADELTA learning algorithm [15] while training FCN model also improves segmentation [27].

However, the problem with Retinal Image segmentation and other biomedical segmentation is that there are only very few datasets available for training Neural Networks. For CNN and FCN, a large dataset is usually considered for training purpose to get optimal results. Nevertheless, that is not the case when it comes to image segmentation applications in Retinal Image segmentation. Very few images and datasets are available from sources like DRIVE and STARE.

A general convolutional neural network focuses on image recognition, with an image as input and a single label as output. However, in biomedical image segmentation, it is necessary not only to determine whether disease exists but also to pinpoint the location of the abnormality. To solve this problem, UNET was introduced by authors of the paper [16] in 2015. UNET is a type of Fast

Convolutional Neural network used to segment biomedical images[16]. The authors of the paper [27] have compared various CNN and FCN models for the application of retinal vessel segmentation. It is suggested that UNET and Dense networks provide better results than other networks.

1.8. UNET :

UNET is based on a more sophisticated architecture known as a "completely convolutional network." This has been updated and modified to function with very few training images and produce more accurate segmentation. The key concept is to add successive layers to a traditional contracting network, replacing pooling operators with upsampling operators. As a result, these layers improve resolution of the output. High resolution features from the contracting path are combined with the upsampling of expanding path to localise [16].
The key concept is to add successive layers to a traditional contracting network, replacing pooling operators with upsampling operators. As a result, the output resolution is improved.

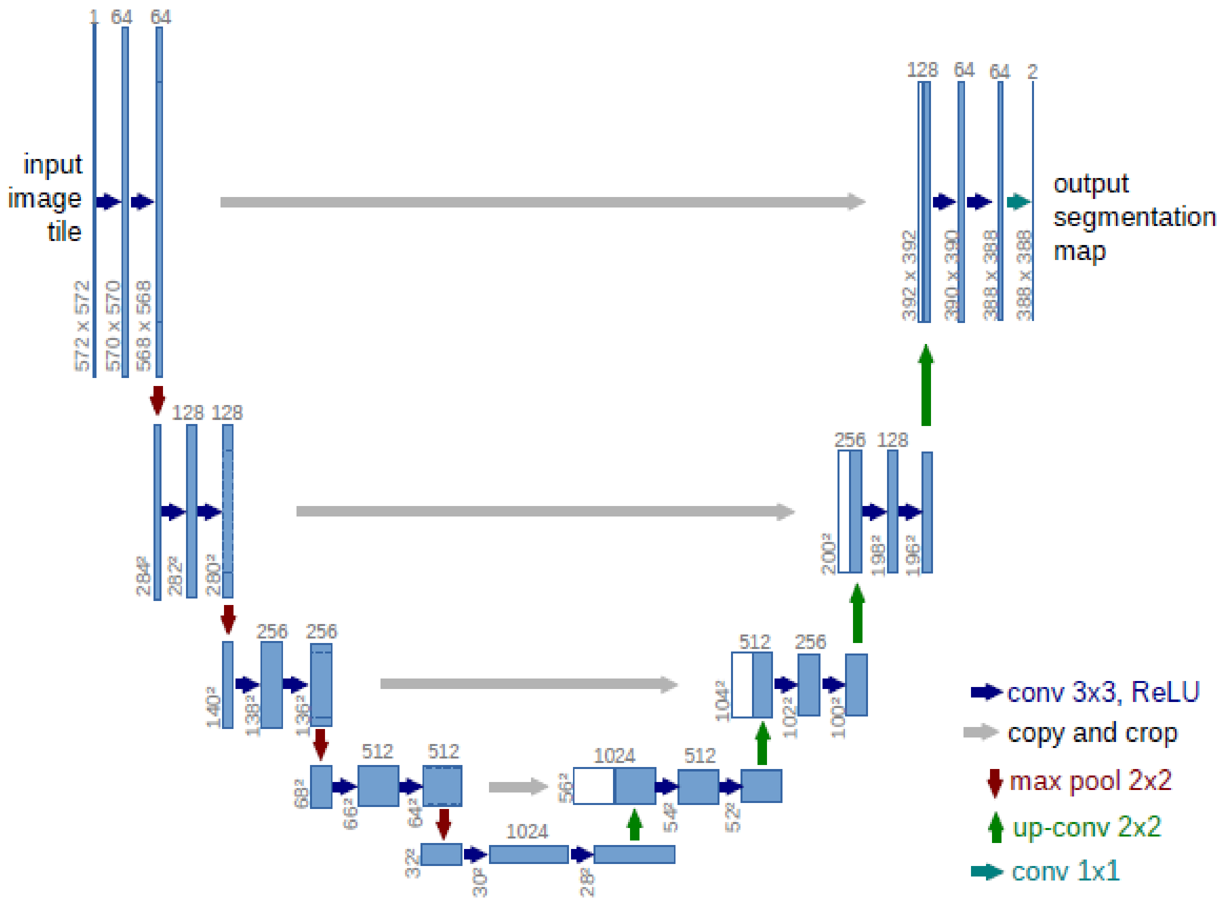


Fig. 10. UNET Architecture[16]

Figure 10 represents the architecture of UNET. The UNET consist of a contracting path to the left referred to as encoder, and an expanding path on the right referred to as decoder, as represented in Figure 10. The proposed UNET is of depth 5. The initial convolutional layer starts with 64 filters and gradually increases the number of filters to 1024 down the contracting path and decreases down to

64 up the expanding path. The output from each layer of downsampling is merged or concatenated with the corresponding upsampling layer. [16]

The proposed UNET was used to segment HeLa cells on glass recorded by DIC. The segmentation results of UNET are shown in figure 11.

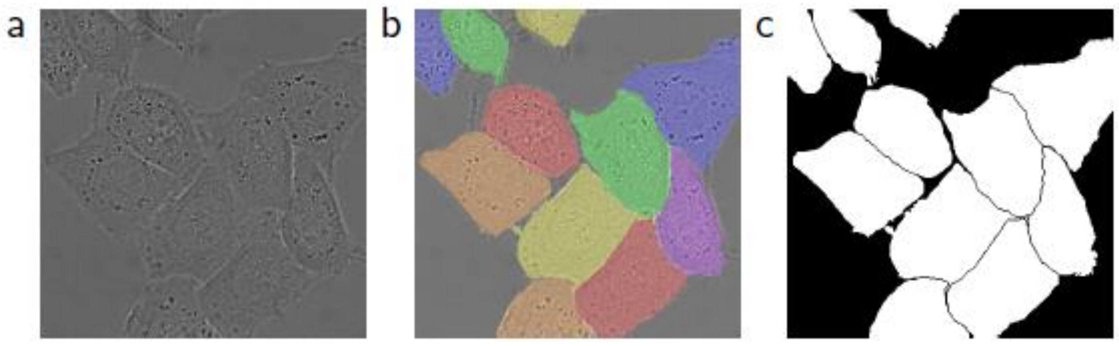


Fig. 21. UNET segmentation of HeLa cells. a) Original cells image, b) Ground Truth of cells, c)UNET Segmentation.

The proposed UNET was tested and compared with other networks at ISIBI cell tracking challenge 2015. The segmentation results (IOU) is shown in Table 1 below.[16]

Table 1. UNET's IOU Segmentation results on the ISBI [16].

Name	**PhC-U373**	**DIC-HeLa**
IMCB - SG	0.2669	0.2935
KTH - SE	0.7953	0.4607
HOUS - US	0.5323	-
Second - best	0.83	0.46
UNET	0.9203	0.7756

1.9. Convolutional Networks Based on UNET :

After the introduction of UNET, many researchers have proposed different architectures based on UNET for multiple biomedical image segmentation applications like UNET++ for cell segmentation [28], where the skip connections in UNET is redesigned. Few of the UNET based architectures were developed for the sole purpose of retinal vessel segmentation, which is proposed by authors of papers [7][10][11][12][8] and [34]. Few of these unique UNET based approaches id discussed in the section below.

1.9.1. AD-UNET :

The Attention-Dense-UNET (AD-UNET) for micro-vessel Image Segmentation was proposed by the authors of the paper [10]. In preprocessing, the image enhancements highlight the retinal blood vessels. The original retinal images are converted to CLACHE images (Figure 12), which is then given as input to AD-UNET. The AD-UNET algorithm adds an attention block and a dense network to the original UNET network. The architecture of AD-UNET is represented in figure 13.

The accuracy of the algorithm is 0.9663 according to test results on the DRIVE images. The sensitivity is 0.8075, the specificity is 0.9814, the AUC is 0.9846, and the F-measures is 0.820 respectively.

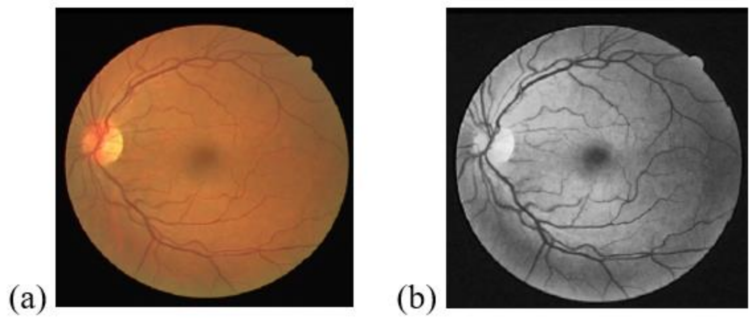


Fig. 12. a)Raw Image b)CLACHE Image

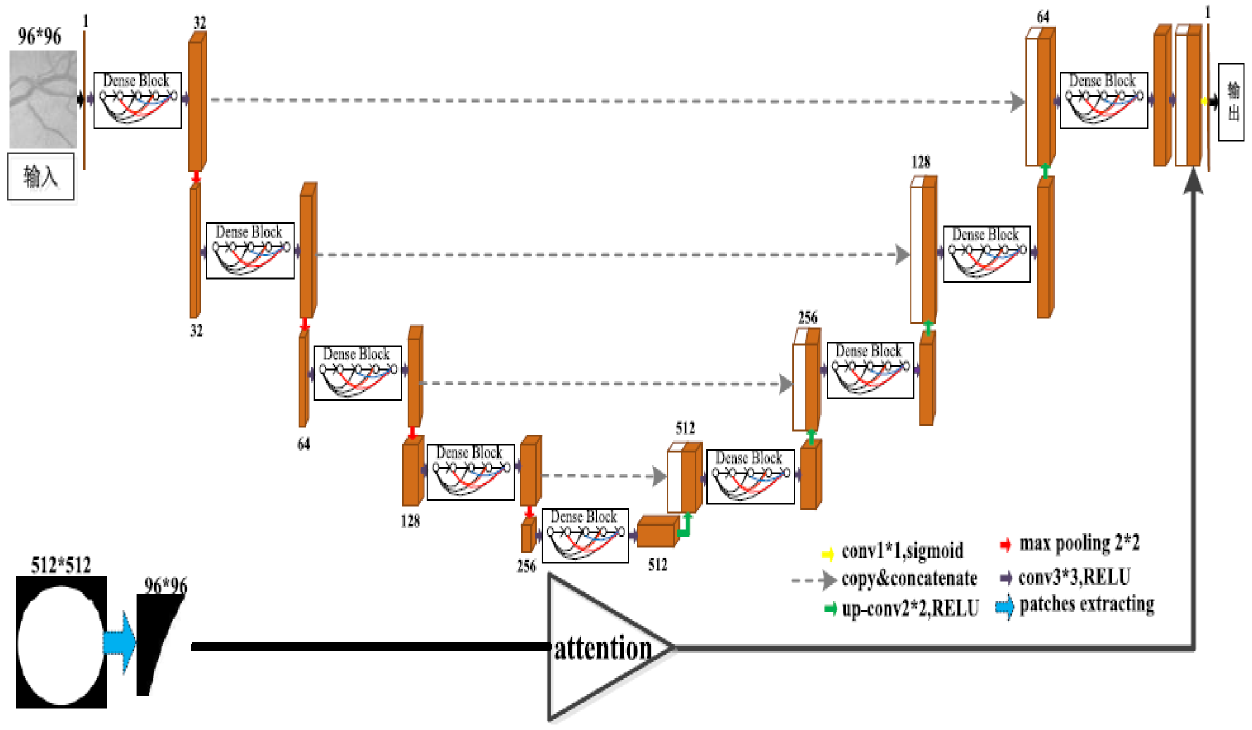


Fig. 13. The architecture of AD-UNET [10].

1.9.2. S-UNET :

In 2019, the authors of the paper [11] suggested S-UNET, a bridge-style Cascade UNET. A Salient UNET (S-UNET) is a bridge UNET architecture with a saliency system, is proposed based on the minimal UNET (Mi-UNET) model, which is a UNET with significantly reduced parameter count. S-UNET implements a cascading strategy in which one net block‘s foreground features are used as the foreground attention for the next UNET block. This cascading S-UNET results in improved input and the inheritance of previous netblock‘s learning experience. Figure 14 depicts the architecture of the S-UNET.

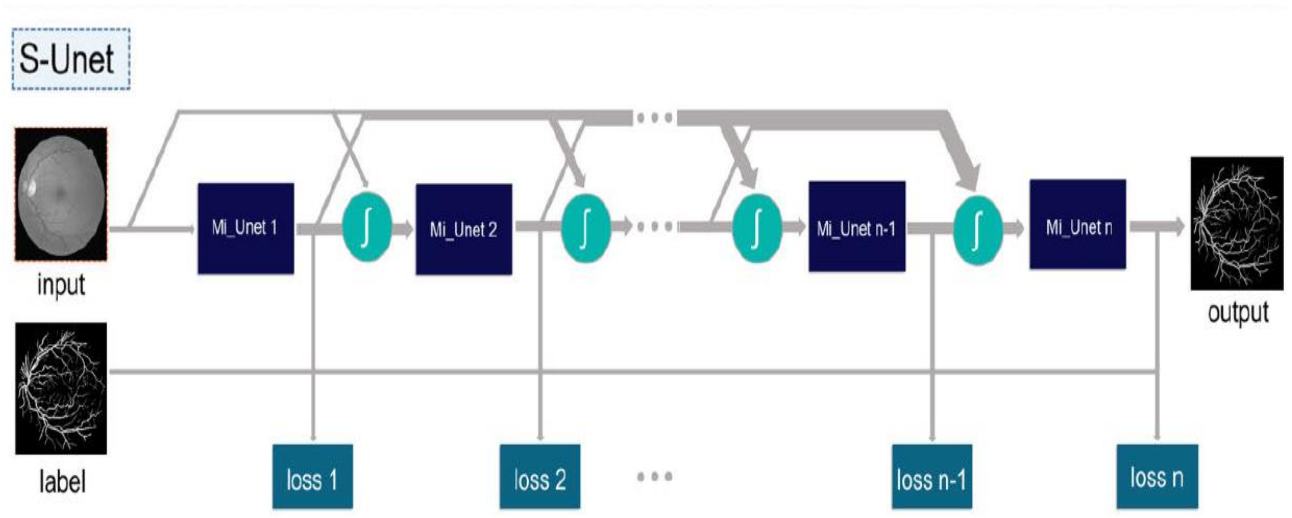


Fig. 14. S-UNET Architecture [11].

1.9.3. GNET :

A FCN model named Gaussian net (GNET) was proposed in 2019 for retinal vessel segmentation by authors of the paper [12]. This model is combined with a saliency model. The proposed FCN Model resembles Gaussian distribution curve hence the name GNET. A saliency detection is used in preprocessing along with greyscale conversion of retinal images that helps to highlight the blood vessels. The saliency image (figure 15) is given as the GNET model’s input. The GNET model has a symmetrical arrangement on both sides. The first layer in the left structure is upsampling, while the other layers are downsampling. The GNET architecture is represented in figure 16. The arrow pointing up represents upsampling; the arrow pointing down represents downsampling; the dotted line represents concatenation. The final layer in the correct system is downsampling, while the other layers are upsampling.

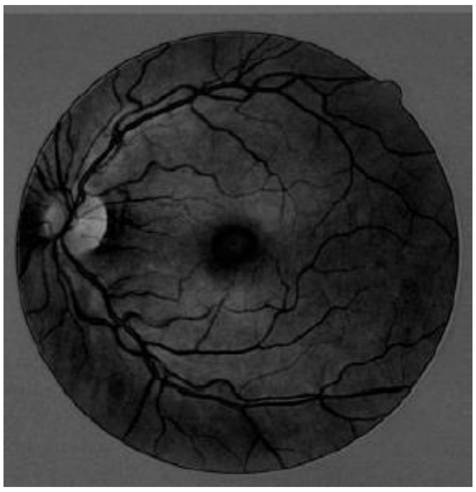


Fig. 15. Saliency Image [12].

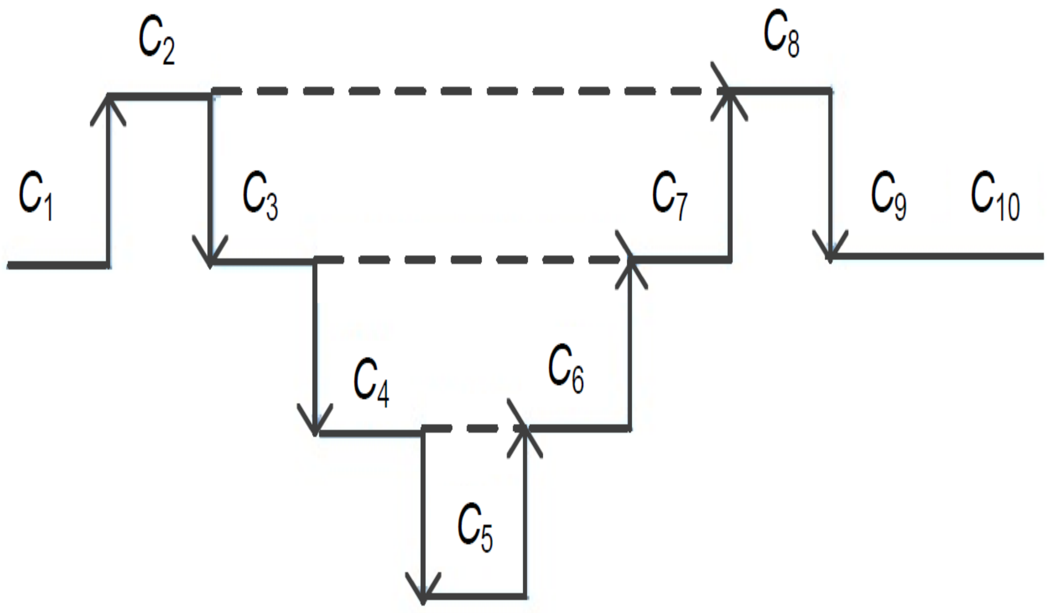


Fig. 16. GNET Architecture [12].

2. Methodology

2.1. Process Methodology :

The project process methodology is represented in figure 17 below.

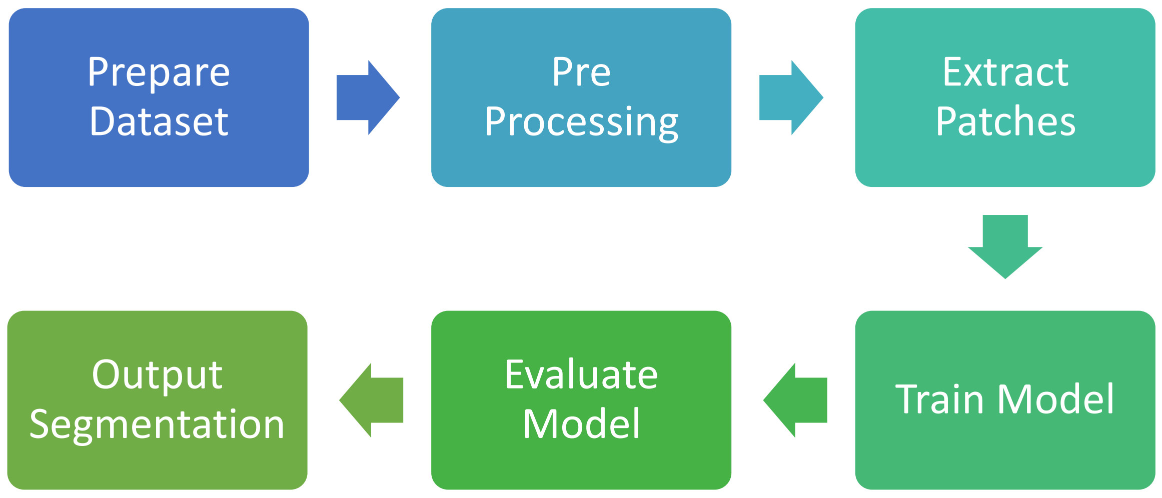


Fig. 17. Methodology

The methodology consist of the following process :

1. Prepare dataset :

   The dataset is in the form of an image, and it should be converted to a suitable format for the program to read the images. In this case, the original image and corresponding ground truth are converted to hdf5 format. HDF stands for Hierarchical Data Format and refers to a group of file formats (HDF4, HDF5) that are used to store and organise large volumes of data.

2. Pre-Processing

   The image dataset is in RGB colour format. It is in best practice to convert the RGB to Greyscale format. It is common practice for RGB image to be converted to greyscale image in image processing because, in a grayscale image, the only colours used are grayscale shades. The explanation for the distinction between such images and every other type of colour image is that each pixel requires only fewer details. The original RGB image and Greyscale converted image from DRIVE database is shown in figure 18.

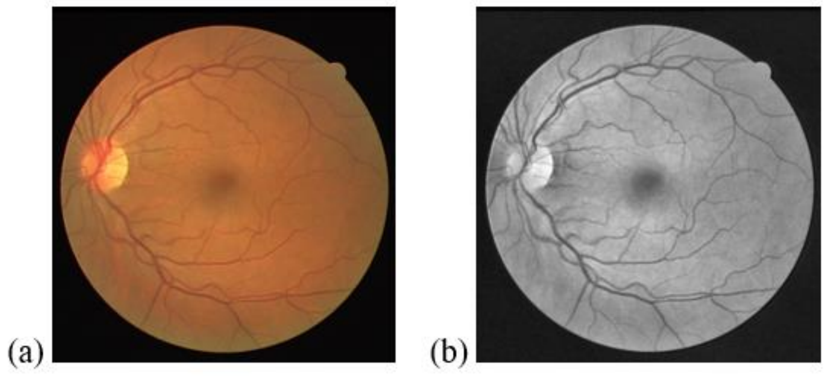


Fig. 18. (a) Original RGB retinal Image. (b) Greyscale retinal Image

3. Extract Patches

   For the model to train, a sample set of patches are extracted from the preprocessed data and corresponding ground truth data. In this case, a sample set of 20000 patches with patch sizes 48/64px are extracted randomly from the data for training the model.

4. Train Model

   The extracted patches are used to train the UNET model.

5. Evaluate Model

   The trained model is then evaluated on the test images to produce segmentations and evaluate segmentation results.

6. Output Segmentation

   The Segmented images are then visualised in image format as shown in figure 19 for interpretation.

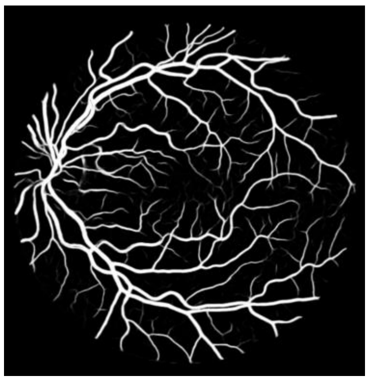


Fig. 19. Segmentation generated by UNET.

2.2. Methodology for Model Optimisation :

The UNET optimisation methodology followed in this project is represented in figure 20 below.

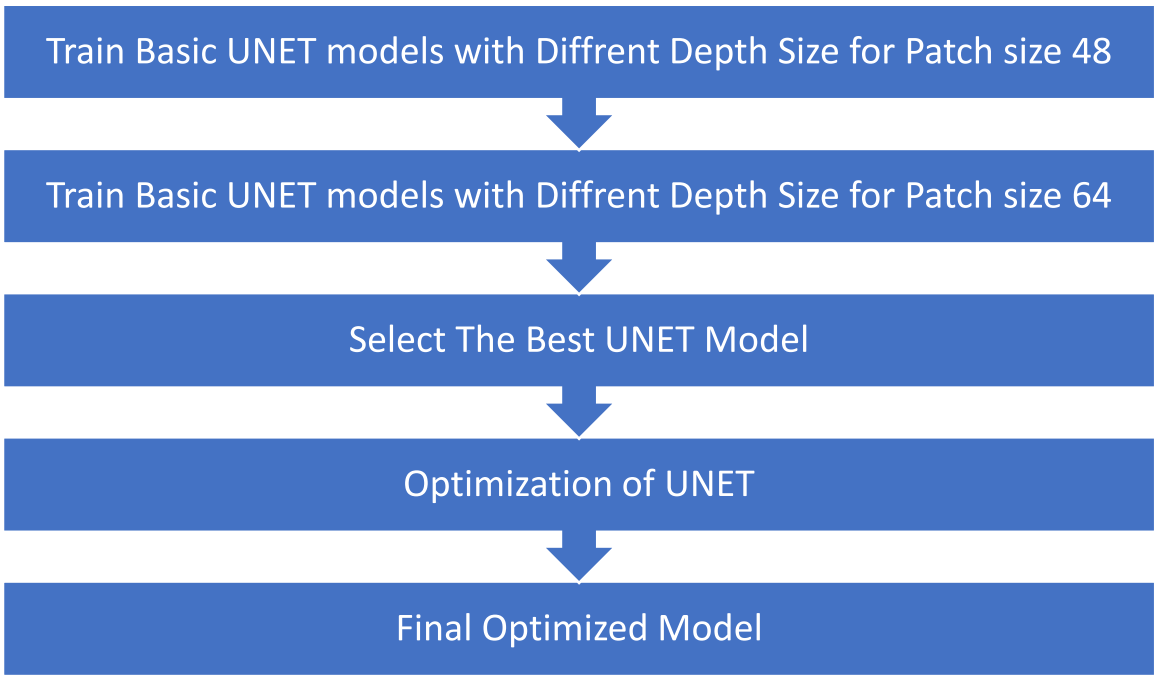

Train Basic UNET models with Diffrent Depth Size for Patch size 48
Train Basic UNET models with Diffrent Depth Size for Patch size 64
Select The Best UNET Model
Optimization of UNET
Final Optimized Model

Fig. 20. Methodology for UNET optimisation.

The Model Optimisation process is as follows :

1. We train basic UNET models with varying depth from 3 – 5 with a data patch size of 48.
2. We train basic UNET models with varying depth from 3 – 5 with a data patch size of 64.
3. The evaluation results of all the basic UNET networks are compared, and the best UNET model is chosen.
4. Then the chosen UNET model‘s architecture is optimised based on techniques and ideas inspired by other research papers.
5. The best optimised UNET is proposed at the end of the research.

3. Experimental Setup :

3.1. Computer Specifications :

Processor: Intel i7 eighth generation.

RAM: 16 GB DDR4

GPU: Nvidia GEFORCE GTX 1050 - 6 GB

OS: Windows 10 Home.

3.2. Software :

Language: Python

Libraries: Tensorflow, Keras

IDE: Visual Studio

3.3. Dataset :

The research is based on retinal vessel segmentation. Hence DRIVE(Digita Retinal Images for Vessel Extraction) dataset was chosen because it is specifically prepared for vessel extraction. The DRIVE consist of the following data :

Table 2. DRIVE dataset.

DRIVE	Training Data	Testing Data
Images	20	20
Ground Truth 1	20	20
Ground Truth 2	-	20
Mask	20	20

3.4. Training :

The training dataset is prepared by extracting 20000 patches of size 48/64px from the greyscale converted retinal images. The training dataset is then divided into two parts: training data and validation data. Two per cent of the training data is used for validation purpose. The dataset reserved for validation is not used in training the model. All the UNET models in this research were trained for 16 EPOCH with a batch size of 8.

4. Evaluation Parameters :

All the evaluation parameters, such as Accuracy, Sensitivity, Specificity, Precision, are all based on algorithm prediction of the positive and the negative cases compared to the actual positive and negative cases. These are classified into four possible cases :

1. **True Positive (TP):** These are cases in which the algorithm predicted 'Positive'.
2. **True Negative (TN):** These are cases in which the algorithm predicted ,'Negative'.
3. **False Positive (FP):** These are the cases in which the algorithm predicted ,'Positive', but they are actually ,'Negative'.
4. **False Negative (FN):** These are the cases in which the algorithm predicted ,'Negative', but they are actually ,'Positive'.

The following evaluating parameters are used to evaluate the UNET models trained in this research.
Sensitivity/Recall:

Sensitiviti or Recall is defined as the proportion of correctly identified positive cases to all the actual positive cases.

$$\text{Sensitivity/Recall} = \frac{TP}{TP+FN} \tag{1}$$

Specificity:

Specificity is defined as the propotion of correctly identified negative cases to all the actual negative cases.

$$\text{Specificity} = \frac{TN}{TN+FP} \tag{2}$$

Precision :

Precision shows the propotion of positive identification cases, which was actually positive.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{3}$$

Accuracy:

Accuracy is the total number of positive and negative cases correctly detected.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FN+FP} \tag{4}$$

F1 Score:

F-measure or F1-score is the harmonic mean of precision and recall. But this measure might not be very helpful when the cases of actual positive and actual negative are imbalanced.

$$\text{F1} = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{5}$$

<u>Matthews correlation coefficient (MCC) :</u>

The MCC considers TP, TN, FP, and FN, hence MCC coefficient is a balanced metric that can be used even though the number of classes are of very different sizes or imbalanced. The MCC is a correlation coefficient that returns a value between -1 and +1 for binary classifications that are observed and predicted. A coefficient of +1 denotes that a models prediction is perfect, a coefficient of 0 denotes that a models prediction is random, and a coefficient of 1 denotes complete disagreement between the model's prediction and the model's observation.

$$\text{MCC} = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{6}$$

<u>AUC ROC:</u>

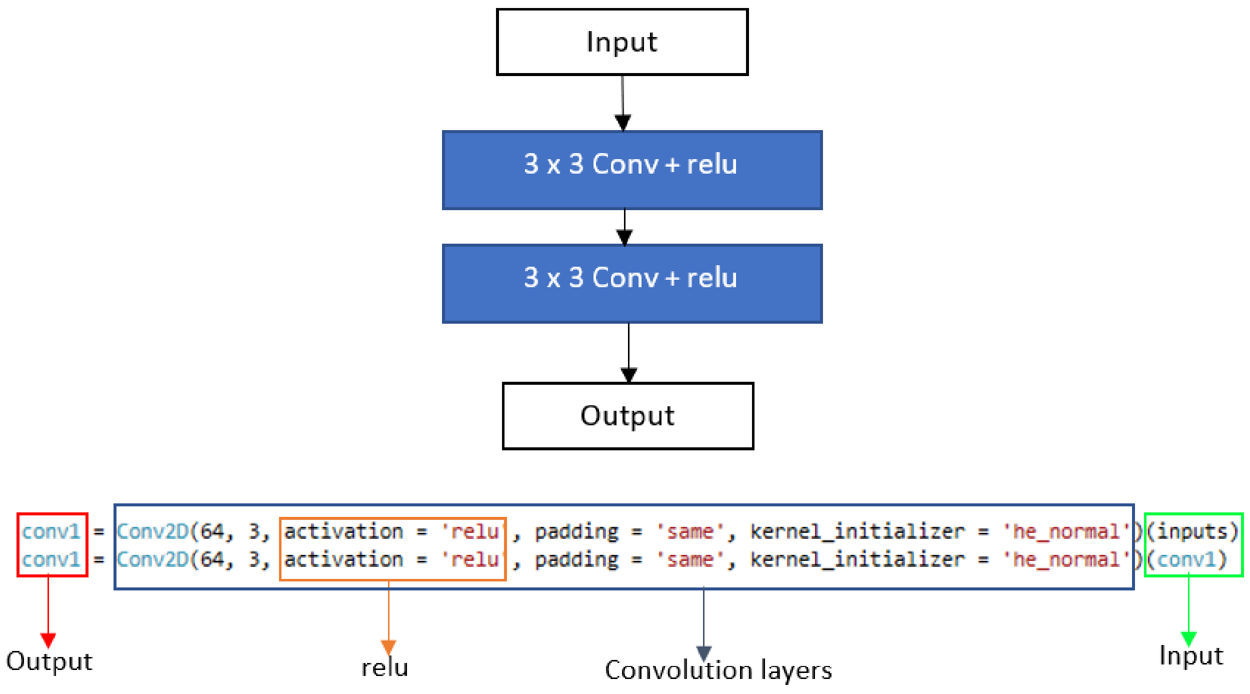
The Receiver Operator Characteristic curve or simply called as ROC curve is a metric for evaluating binary classification problems. It's a probability curve that compares the True Positive Rate (TPR) to the False Positive Rate (FPR) at various threshold values, essentially distinguishing the signal from the noise. The ROC curve that tests a classifier's ability to distinguish between classes is known as the Area Under the Curve (AUC). The AUC_ROC shows the ability of the model to distinguish between positive and negative cases. The higher the AUC means that the model is very well able to distinguish between the positives and the negatives..

5. UNET Model Blocks Overview :

The UNET models developed and tested in this project consist of mainly five functional blocks, namely: Convolution, MaxPooling, Skip connections, Cross Skip Connection and, Up-Sampling.

5.1. Convolution :

In this project, a 3x3 convolution layer is used with rectified linear activation function (relu). If the input is positive, the ‘relu‘ is a piecewise linear function that outputs it directly. Otherwise, it would return a value of zero. The convolution layer with window size is 3x3 is used in this project. The convolution layer generates a tensor of outputs by convolving the layer input with a convolution kernel. The convolution block and its keras implementation used to develop UNET models is illustrated in Figure 21. In the keras implementation, the input and output variables are highlighted in green and red colour boxes. The convolutional layer and its relu activation are highlighted in blue and orange colour boxes. These blocks are used in encoder and decoder parts of the UNET model.

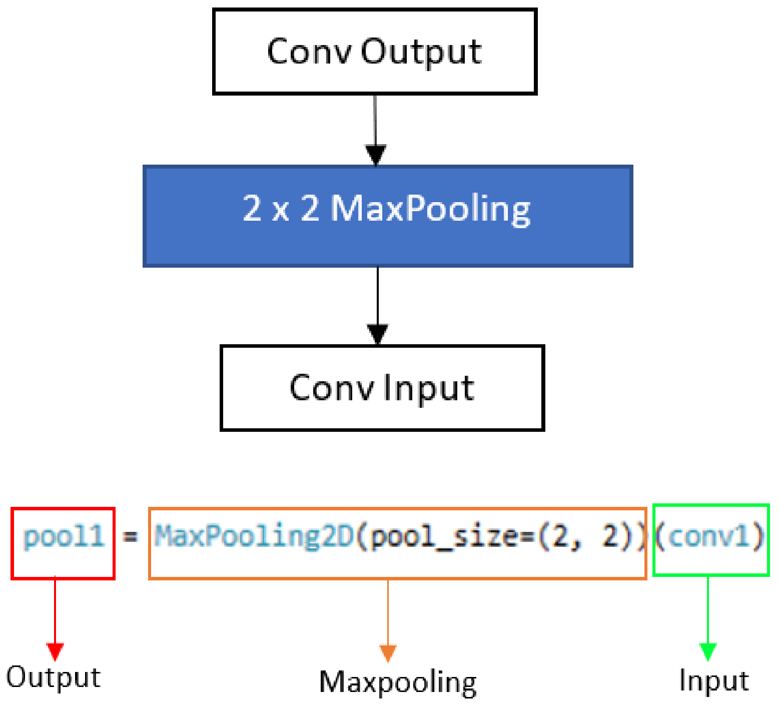


```
conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
```

Fig. 21. Convolution Block and its Implementation in keras.

5.2. Pooling :

Pooling is a technique used in UNET for downsampling. The downsampling is done based on the specified pool size or window. In this project, 2 x 2 MaxPooling is used for downsampling in the encoder path to pass data from the output of the higher convolution block down to the input of the lower convolution block. The pooling block and its keras implementation are illustrated in Figure 22. Here 2 x 2 is the pool size for downsampling. In the keras implementation, the convolution output given as input to the pooling block is hilighted in green colour box and the output variable used as the input for another convolutional block is highlighted in red colour box. The max pooling operation is highlighted in orange colour box.

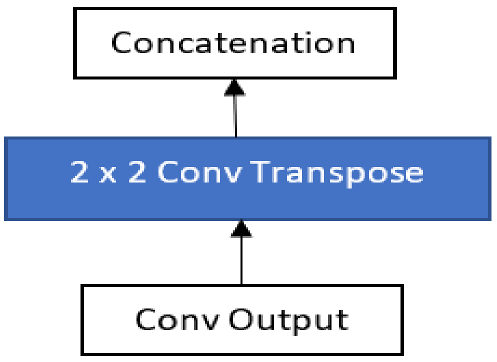


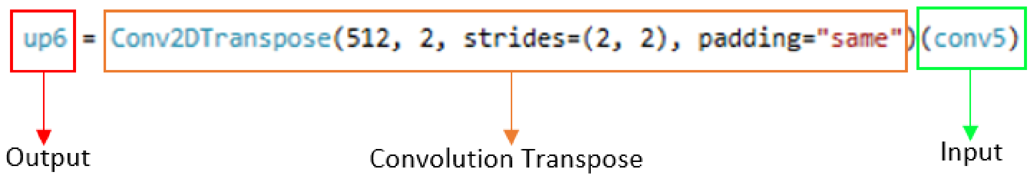
```
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
```

Fig. 22. Pooling Block and its Implementation in keras.

5.3. Up Sampling :

Upsampling is used in the decoder path of UNET for passing data from lower convolution block to higher convolution blocks for concatenation with skip connection. A 2 x 2 Convolution Transpose is used for upsampling in this project, where 2 x 2 is the window size of the transposed convolution. Transposed convolutions are commonly used when a transformation in the reverse direction of a regular convolution is desired. The Up Sampling and its keras implementation illustrated in Figure 23. In the keras implementation, the output from convolutional block, which acts as the input to the upsampling, is highlighted in green colour box, and the output of upsampling which is given as input to another convolutional block, is highlighted in red colour box. The convolution transpose operation in keras is highlighted in orange colour box.



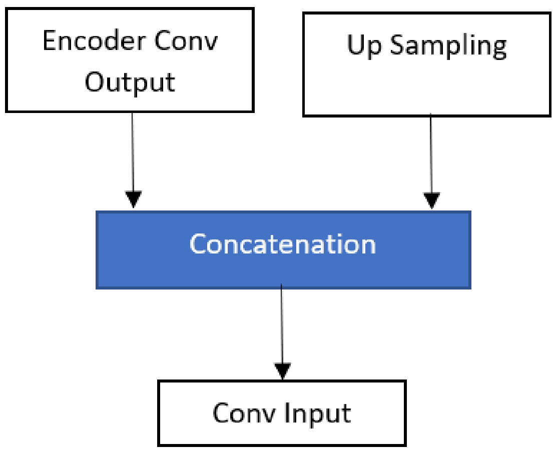


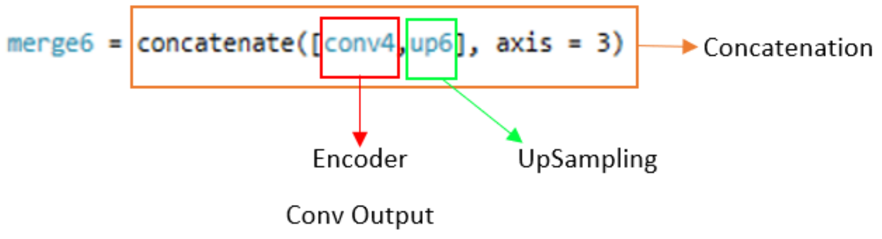
```
up6 = Conv2DTranspose(512, 2, strides=(2, 2), padding="same")(conv5)
```

Fig. 23. Up Sampling and its Implementation in keras.

5.4. Skip Connection :

Since the skip connection in UNET is from convolution block of encoder path to a convolution block of decoder path with exact filter sizes, it can be directly achieved by concatenation operation by simply calling the output of the desired convolution block from the encoder and concatenating it with the output of the upsampling convolution. This is then given as input to the desired convolution block of the decoder path. This operation and its keras implementation are illustrated in Figure 24. In the keras implementation, the upsampling and the convolution output variables are highlighted in green and red colour boxes. The concatenation operation in keras is highlighted in orange colour box.



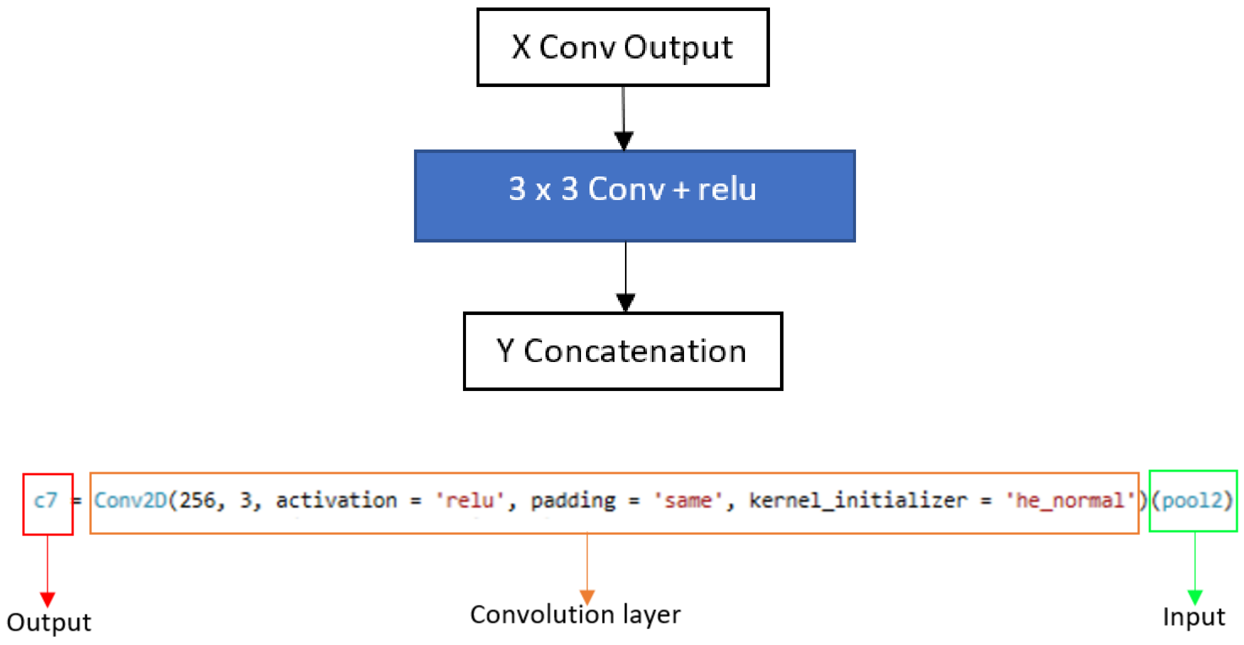


```
merge6 = concatenate([conv4,up6], axis = 3)
```

Fig. 24. Concatenation Operation and its Implementation in keras.

5.5. Cross skip Connection:

The cross skip connection is proposed in this project for passing data for concatenation with decoder path convolution block from encoder path convolution block. This cross skip connection is used to connect convolution block with different filter sizes. In this project, a simple 3 x 3 Convolution with activated relu is used for cross skip connections. The Cross Skip Connection and its keras implementation are illustrated in figure 25, where X denotes the filter size of the convolution output, and Y denotes the filter size of the concatenation. In the keras implementation, the input and the output variables are highlighted in green and red colour boxes. The convolutional layer in keras in highlighted in orange colour box.



```
c7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
```

Fig. 25. Cross Skip Connection and its Implementation in keras.

6. Experiment:

In this chapter basic structure of UNET with varied depths is built and tested for RBVS. Then the best performing basic UNET is selected for further improvisation and modifications.

6.1. UNET with Depth 3 :

The UNET with depth 3 with initial filters 64 and 32 was built. The architecture of the model is represented in figure 26.

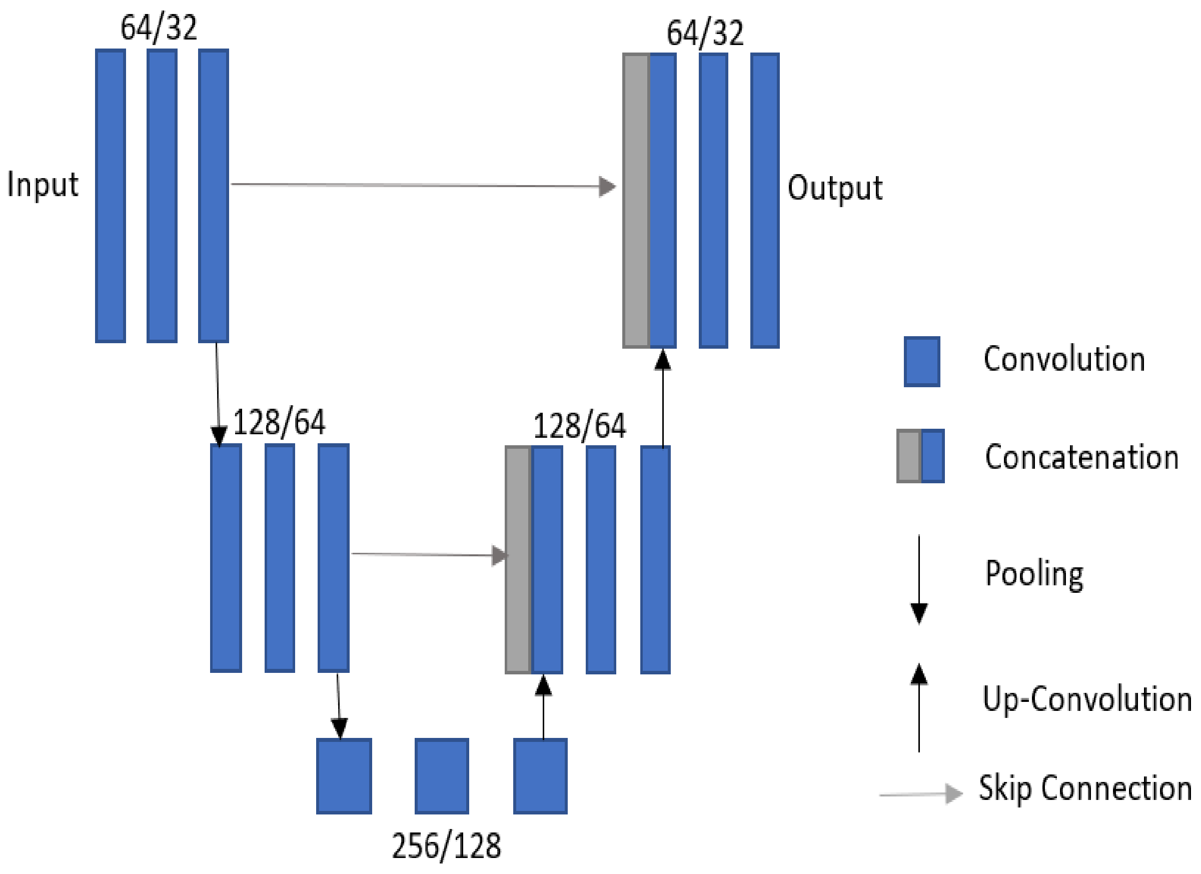


Fig. 26. UNET Depth 3 Architecture

The UNET depth 3 consists of an encoder or contracting path on the right side and a decoder or expanding path on the left side. Two variants of UNET is designed for experimental purposes. Variant 1 consist of filters 64, 128, 256 at encoder and 256, 128, 64 at the decoder. Variant 2 consists of filters 32, 64, 128 at encoder and 128, 64, 32 at the decoder.

The two variants of UNET with depth three were trained with a data patch size of 48 and 64. The evaluated results are shown in Table 3 below.

Table 3. Evaluation results of UNET 3 variants.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISION	ACCURACY	AUC-ROC
PATCH SIZE 48							

UNET 3 F 32	0.7930	0.8189	0.7767	0.98051	0.8659	0.9522	0.9760
UNET 3 F 64	0.7946	0.8212	0.7879	0.9788	0.8574	0.9522	0.9760
PATCH SIZE 64							
UNET 3 F 32	0.7952	0.8201	0.7710	0.9823	0.8758	0.9529	0.9774
UNET 3 F 64	0.7943	0.8201	0.7793	0.9804	0.8655	0.9524	0.9771

6.2. UNET with Depth 4 :

The UNET with depth 4 with initial filters 64 and 32 was built and the architecture of is as shown in figure 27.

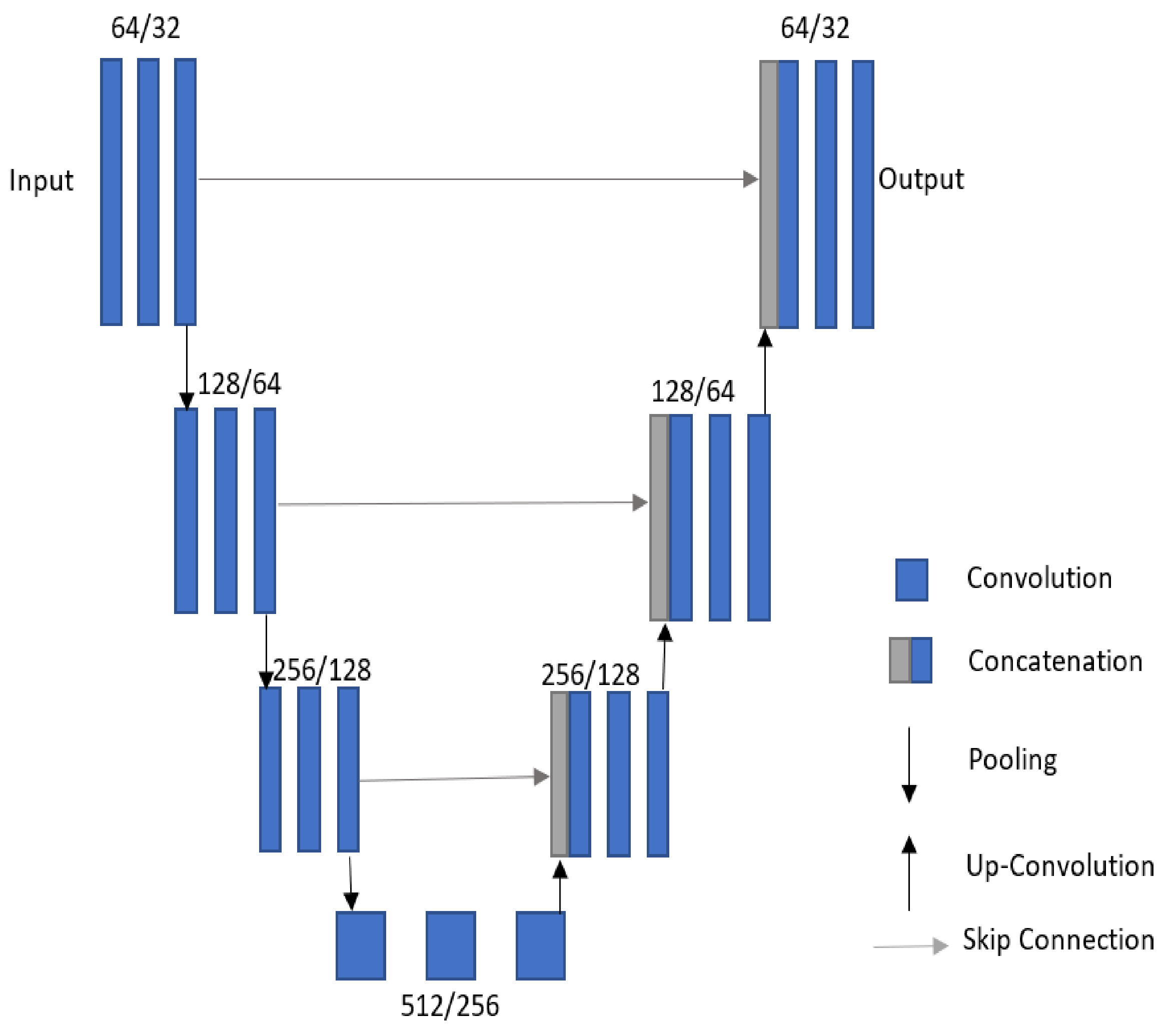


Fig. 27. UNET Depth 4 Architecture

The UNET depth 4 consists of an encoder or contracting path and a decoder or expanding path. Two variants of UNET is designed for experimental purposes. Variant 1 consist of filters 64, 128, 256, 512 at encoder and 512, 256, 128, 64 at decoder. Variant 2 consist of filters 32, 64, 128, 256 at encoder and 256, 128, 64, 32 at decoder.

The two variants of UNET with depth four were trained with a data patch size of 48 and 64. The evaluated results are shown in Table 4 below.

Table 4. Evaluation results of UNET 4 variants.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISION	ACCURACY	AUC-ROC
PATCH SIZE 48							
UNET 4 F 32	0.7890	0.8152	0.7712	0.9804	0.8646	0.9513	0.9750
UNET 4 F 64	0.7893	0.8148	0.7652	0.9817	0.8712	0.9516	0.9753
PATCH SIZE 64							
UNET 4 F 32	0.7806	0.8040	0.7332	0.9853	0.8899	0.9502	0.9743
UNET 4 F 64	0.7905	0.8158	0.7651	0.9821	0.8736	0.9519	0.9752

6.3. UNET with Depth 5 :

The UNET with depth 5 with initial filters 64 and 32 was built and the architecture is as shown in figure 28.

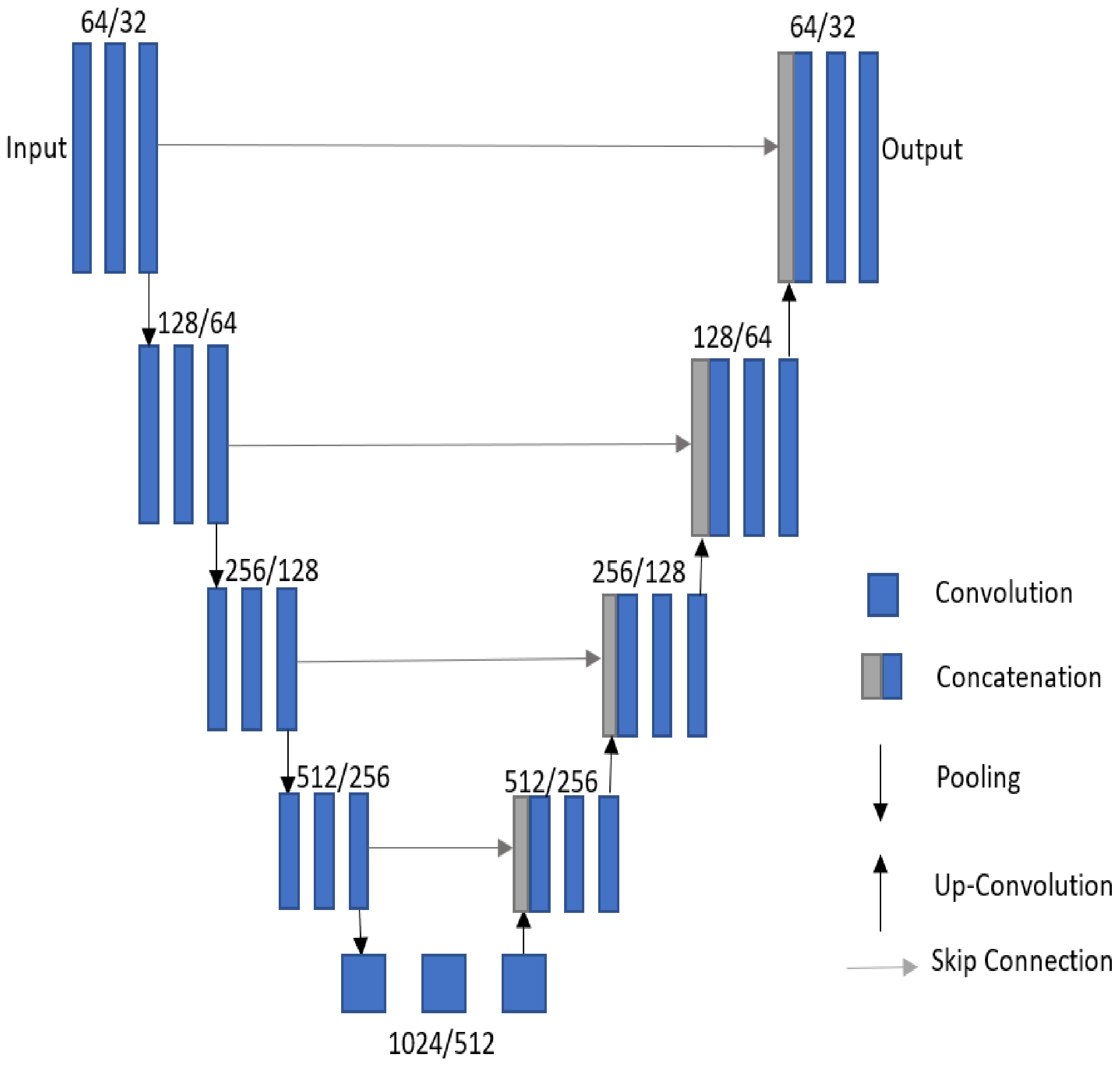


Fig. 28. UNET Depth 5 Architecture.

The UNET Depth 5 consists of an encoder or contracting path and a decoder or expanding path. Two variants of UNET is designed for experimental purposes. Variant 1 consist of filters 64, 128, 256, 512, 1024 at encoder and 1024, 512, 256, 128, 64 at decoder. Variant 2 consist of filters 32, 64, 128, 256, 512 at encoder and 512, 256, 128, 64, 32 at decoder.

The two variants of UNET with depth five were trained with a data patch size of 48 and 64. The evaluated results are shown in Table 5 below.

Table 5. Evaluation results of UNET 5 variants.

NETWORK	**MCC**	**F1**	**SENSITIVITY**	**SPECIFICITY**	**PRECISION**	**ACCURACY**	**AUC-ROC**
PATCH SIZE 48							
UNET 5 F 32	0.7939	0.8218	0.8051	0.9750	0.8391	0.9514	0.9760

UNET 5 F 64	0.7967	0.8243	0.8094	0.9750	0.8397	0.9520	0.9773
PATCH SIZE 64							
UNET 5 F 32	0.7900	0.8152	0.7636	0.9822	0.8742	0.9518	0.9760
UNET 5 F 64	0.8005	0.8272	0.8073	0.9766	0.8482	0.9531	0.9784

6.4. Discussion :

6.4.1. Comparison :

The best of the basic networks with different depth, initial filters and patch size is compared in this section.

Legend :

N(3,4,5) – Depth of UNET model.

F – Initial Filters.

PS – Patch Size.

Table 6. Evaluation results of best of UNET 3, 4 & 5 variants.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISION	ACCURACY	AUC-ROC
UNET 3 F 32 PS 64	0.7952	0.8201	0.7710	0.9823	0.8758	0.9529	0.9774
UNET 4 F 64 PS 64	0.7905	0.8158	0.7651	0.9821	0.8736	0.9519	0.9752
UNET 5 F 64 PS 64	0.8005	0.8272	0.8073	0.9766	0.8482	0.9531	0.9784

From table 6 above, we see that the UNET with depth 5, initial filters 64 and data with a patch size of 64 for training and evaluation (UNET 5 F 64 PS 64) has the best MCC score. This model also shows that it has better accuracy and AUC-ROC compared to other models. From these evaluation results, the UNET with depth 5 and initial filters 64 which is trained on data with patch size of 64 (UNET 5 F 64 PS 64), is efficient in retinal vessel segmentation. This model is used as a base reference for further improvisation of the UNET algorithm.

Figures 29, 30 and 31 illustrate the AUC-ROC of UNET 3 F 32 PS 64, UNET 4 F 64 PS 64 and UNET 5 F 64 PS 64.

X – Axis = False Positive Rate (FPR)

Y – Axis = True Positive Rate (TPR)

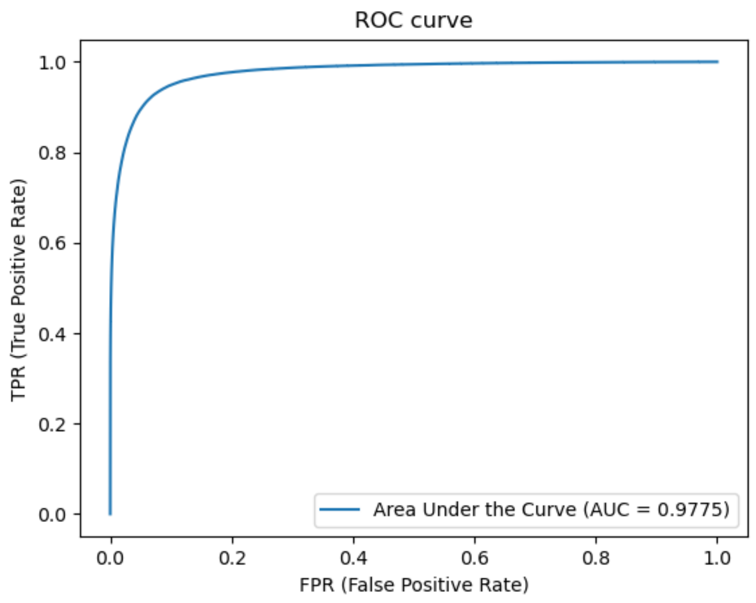


Fig. 29. AUC – ROC of UNET 3 F 32 PS 64.

X – Axis = False Positive Rate (FPR)

Y – Axis = True Positive Rate (TPR)

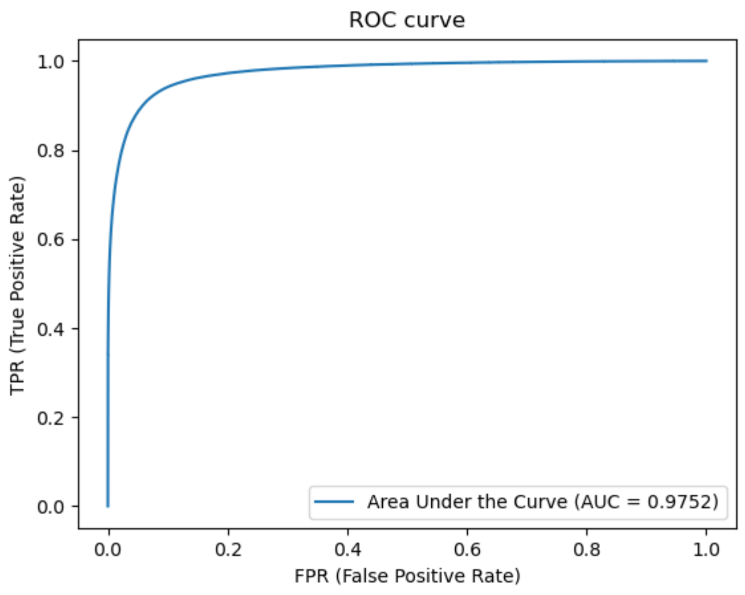


Fig. 30. AUC – ROC of UNET 4 F 64 PS 64.

X – Axis = False Positive Rate (FPR)

Y – Axis = True Positive Rate (TPR)

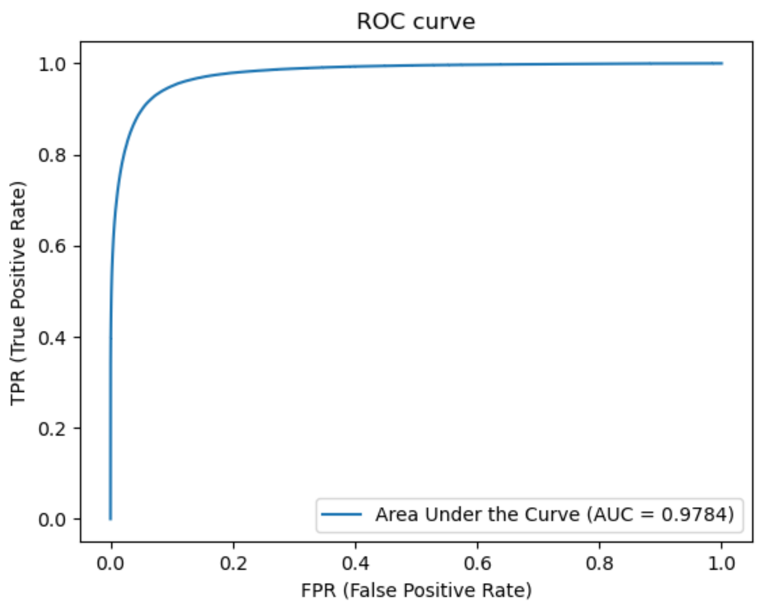
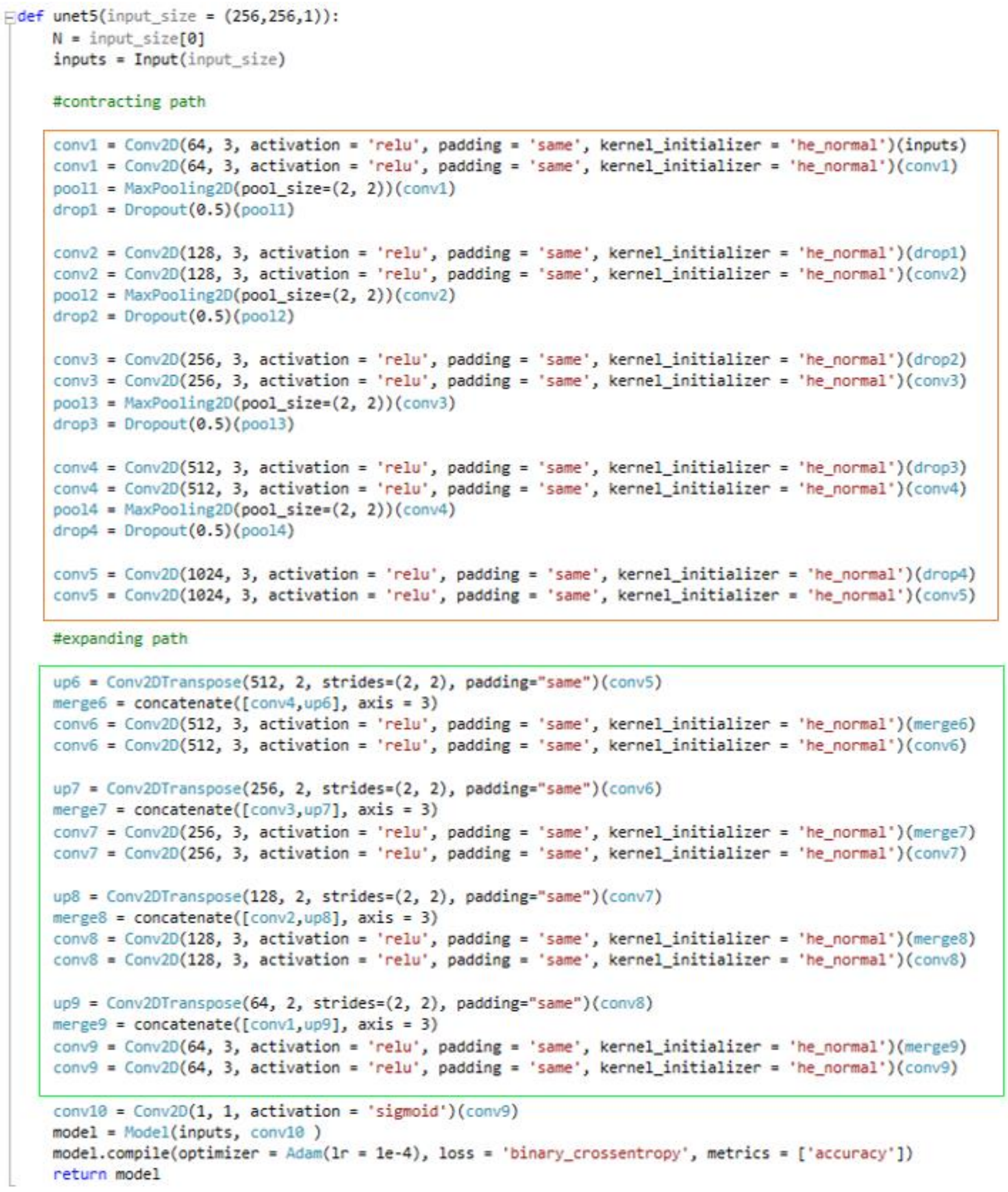


Fig. 31. AUC – ROC of UNET 5 F 64 PS 64.

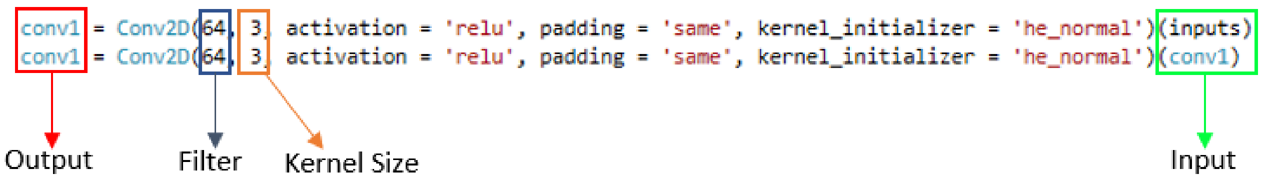
6.4.2. UNET 5 F 64 PS 64 Code :



```
def unet5(input_size = (256,256,1)):
    N = input_size[0]
    inputs = Input(input_size)

    #contracting path

    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    drop1 = Dropout(0.5)(pool1)

    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    drop2 = Dropout(0.5)(pool2)

    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    drop3 = Dropout(0.5)(pool3)

    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)
    drop4 = Dropout(0.5)(pool4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop4)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv5)

    #expanding path

    up6 = Conv2DTranspose(512, 2, strides=(2, 2), padding="same")(conv5)
    merge6 = concatenate([conv4,up6], axis = 3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2DTranspose(256, 2, strides=(2, 2), padding="same")(conv6)
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)

    up8 = Conv2DTranspose(128, 2, strides=(2, 2), padding="same")(conv7)
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv8)

    up9 = Conv2DTranspose(64, 2, strides=(2, 2), padding="same")(conv8)
    merge9 = concatenate([conv1,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)

    conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)
    model = Model(inputs, conv10 )
    model.compile(optimizer = Adam(lr = 1e-4), loss = 'binary_crossentropy', metrics = ['accuracy'])
    return model
```

Fig. 32. UNET 5 F 64 PS 64 code.

The code used for UNET 5 F 64 PS 64 is shown in figure 32. The contracting path of the UNET is highlighted by orange colour box, and the expanding path of the UNET is highlighted by green colour box. The contracting path consists of five convolutional blocks, including the base convolutional block represented as conv1, conv2, conv3, conv4 and conv5. The expanding path consists of four convolutional blocks represented as conv6, conv7, conv8, conv9. Each convolutional block holds the parameters of its own, which is used to process the information it receives. In this code every convolutional layer in a block is given filter size, kernel size, activation, padding and kernel initialiser as represented in figure 33.



```
conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
```

Fig. 33. Convolutional block code.

The filter size represents the number of filters in the convolutional layer output. The kernel size represents the height and width of the convolutional window. The activation used in this code is relu, this activation function provides a positive output for positive values or outputs zero otherwise. The padding used is “same“; this keeps the output of the convolutional layer same size as the input. The kernel initialiser “he_normal“ initialises weights based on the inputs. A dropout function is used after each convolutional block in the contracting path, this dropout function sets random input values to zero during training which helps to prevent overfitting.

Max pooling with window size 2x2 is used for downsampling data from one convolutional block to another in the contracting path. A convolutional transpose function that performs reverse convolution is used for upsampling data from one convolutional block to another in the expanding path. At the end of the network, the output is passed through one last convolutional layer with filter size and kernel size of one and with “sigmoid“ activation, this activation returns a value close to zero for values less than -5 and returns a value close to 1 for values greater than 5.

6.4.3. UNET 5 F 64 PS 64 Training Visualisation:

The training accuracy of the UNET 5 F 64 PS 64 is visualised in figure 34. The orange colour line denotes the training accuracy and the blue colour line represents the validation accuracy. From figure 34, we can see that the training accuracy gradually increases as the number of epochs increases and the validation accuracy increases in the first few epochs and gradually reduces in the later epochs; it suggests that the model is losing data during overfitting.

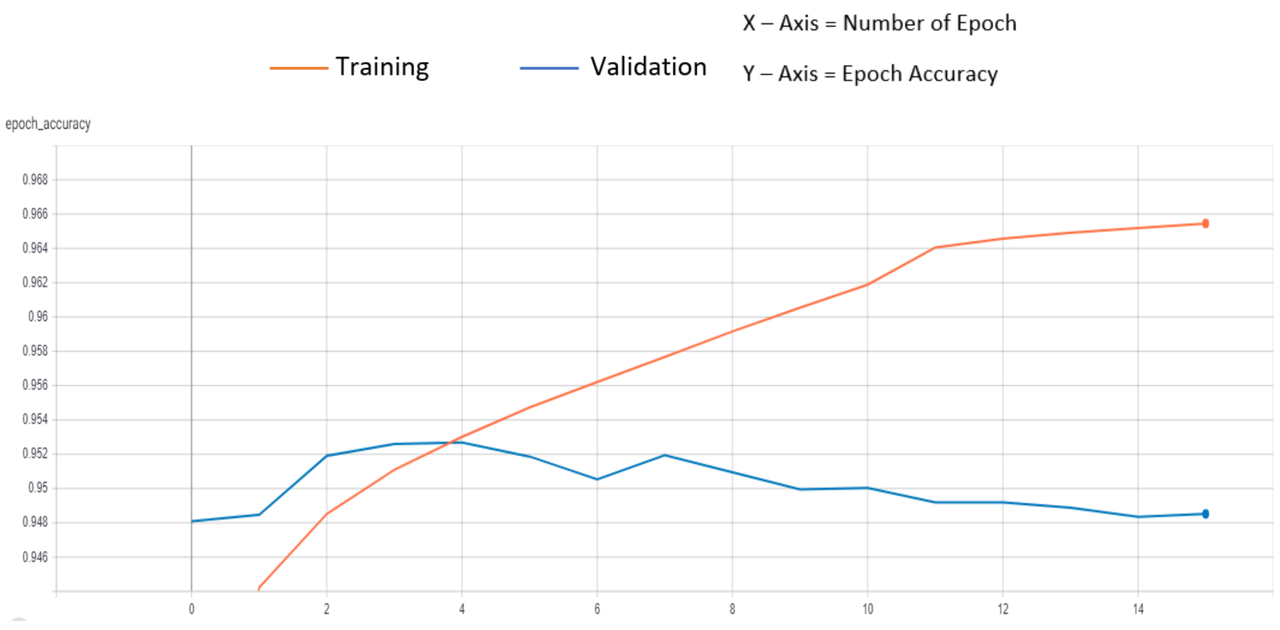


Fig. 34. Training Epoch Accuracy of UNET 5 F 64 PS 64.

6.4.4. Segmentation Results:

The segmentation results of the UNET 5 F 64 PS 64 is shown in figure 35. Figure 35(a) shows the greyscale image, and the ground truth is shown in figure 35(b). The RBVS generated by UNET 5 F 64 PS 64 is shown in figure 35(c). The difference between the ground truth and the segmentation is shown in figure 35(d). In the difference image, the difference is marked in red colour. Few of the difference is due to the human error in ground truth failing to mark very thin blood vessels, few of the difference is due to failure of the UNET to segment thin blood vessels correctly and a majority of the difference is due to the very small difference in the size and of the line used in the ground truth and size of the line in the segmentations generated by UNET. As UNET segmentations are computer-generated, their segmentation lines are exactly the size of the retinal blood vessels. However, the ground truth is human traced segmentation of retinal blood vessels, and the size may differ compared to the actual size.

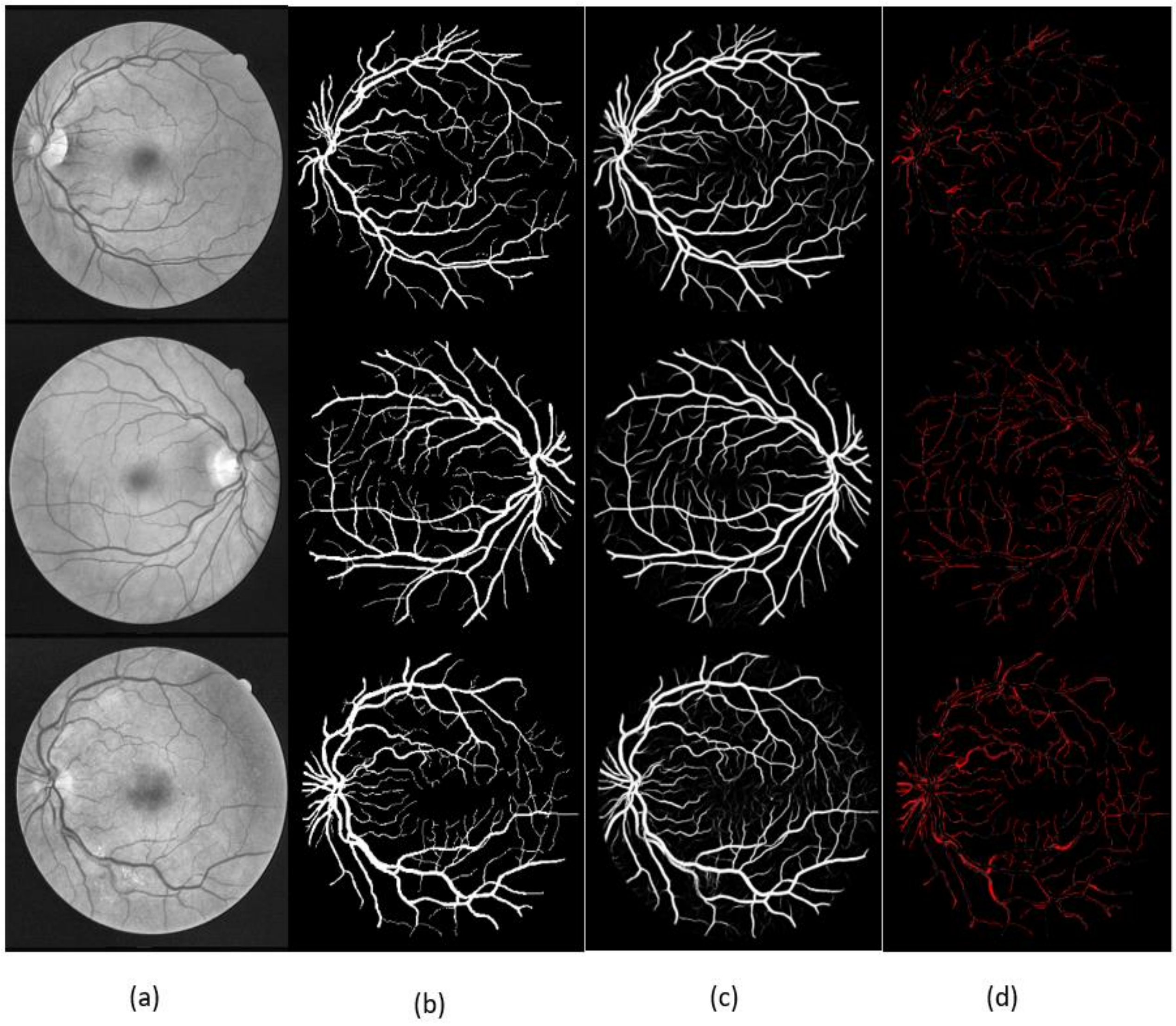


Fig. 35. Segmentation results of UNET 5 F 64 PS 64. (a) Greyscale image, (b) Ground Truth, (c) UNET 5 F 64 PS 64 Segmentation, (d)Difference image.

6.5. Optimisation of UNET :

From the previous chapter on basic UNET models, UNET with depth 5, initial filters 64 and training dataset with patch size 64(UNET 5 F 64 PS 64) showed efficient retinal vessel segmentation results compared to other basic UNET models. In this section, a few modifications and improvements are tested to obtain more efficient retinal vessel segmentation.

The process represented in figure 36 was followed for testing improvements and modifications :

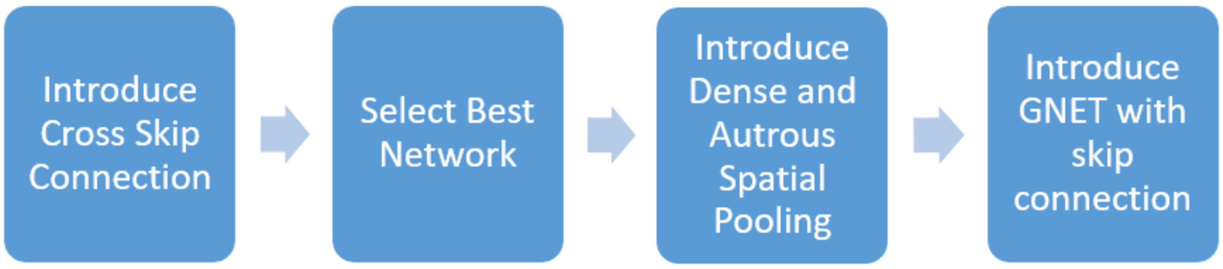

Introduce Cross Skip Connection
Select Best Network
Introduce Dense and Autrous Spatial Pooling
Introduce GNET with skip connection

Fig. 36. UNET modification and improvement process.

The following models were built and tested during the process :

1. UNET with Cross skip connections from the lower level to the higher level (UNET C).
2. UNET with Cross skip connection from the upper level to the lower level (UNET UC).
3. UNET with Cross skip connections from the higher level to the lower level with Dense network and Autrous spatial pooling at the base (UNET UCDA).
4. Gradient UNET with Cross skip connections from the higher level to the lower level with Dense network and Autrous spatial pooling at the base (UNET G UCDA).

6.6. UNET C :

The architecture of UNET C is represented in figure 37. The UNET C consists of a structure of basic UNET 5 F 64. However, the difference is that UNET C consists of extra skip connections from the convolution block of the encoder to the convolution block of the decoder. An extra skip connection is from the lower level convolution block of the encoder to the higher level convolutional block of the decoder, as shown in figure 37. The skip connection from the convolution block of the encoder that is parallel to the convolution block of the decoder and immediate lower level convolutional block of the encoder is concatenated first and then concatenated with the corresponding convolutional block of the decoder.

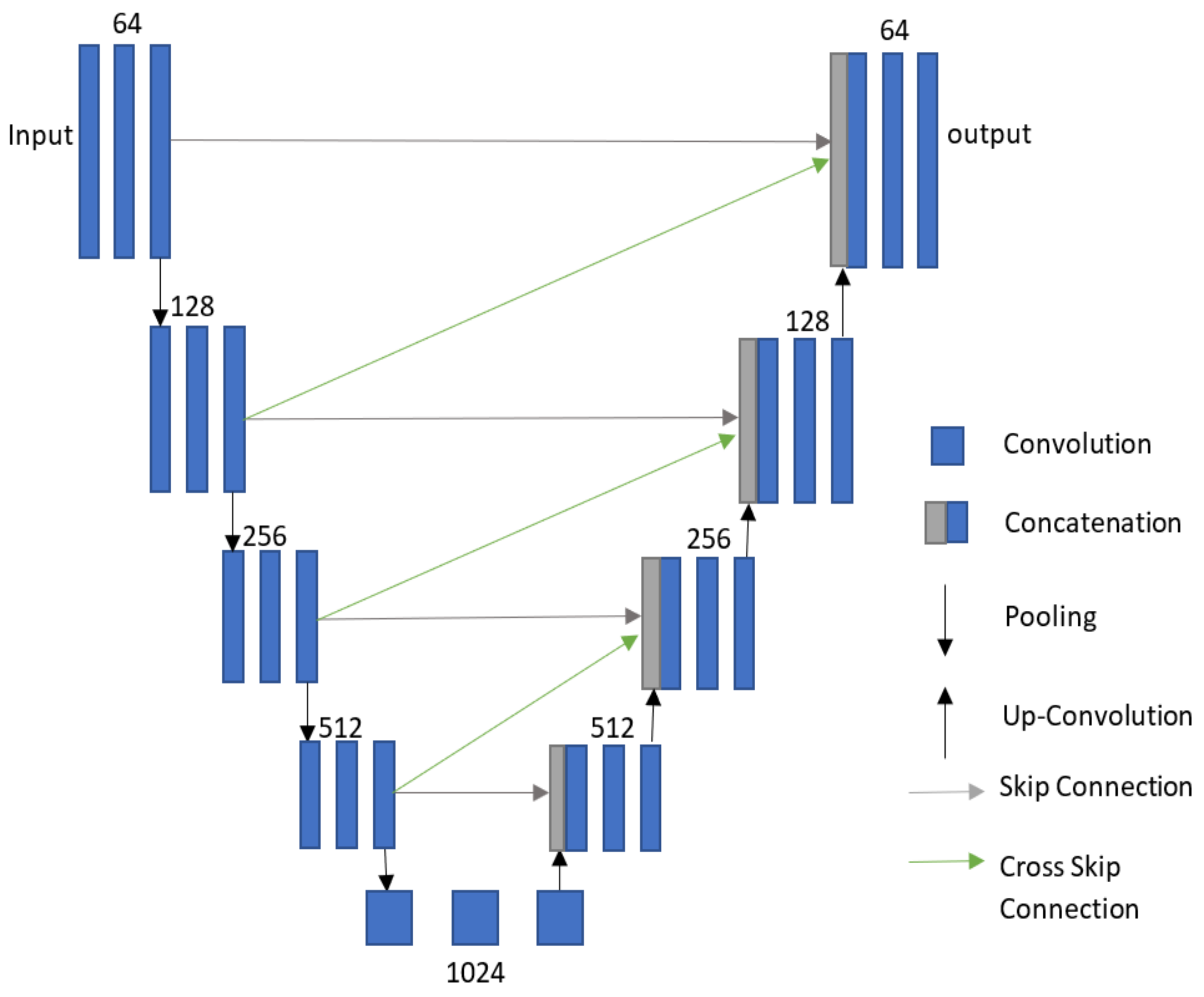


Fig. 37. UNET C architecture.

The UNET C model was trained for 16 epoch and the following results shown in table 7 was obtained during evaluation.

Table 7. Evaluation results of UNET C.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISION	ACCURACY	AUC-ROC
UNET C	0.7929	0.8174	0.7634	0.9831	0.87952	0.9525	0.9776

6.7. UNET UC :

The architecture of UNET UC is represented in figure 38. The UNET UC consists of a structure of basic UNET 5 F 64. However, the difference is that UNET UC consists of extra skip connections from the convolution block of the encoder to the convolution block of the decoder. An extra skip connection is from the higher level convolution block of the encoder to the lower level convolutional block of the decoder, as shown in figure 38. The skip connection from the convolution block of the encoder that is parallel to the convolution block of the decoder and immediate higher level convolutional block of the encoder is concatenated first and then concatenated with the corresponding convolutional block of the decoder.

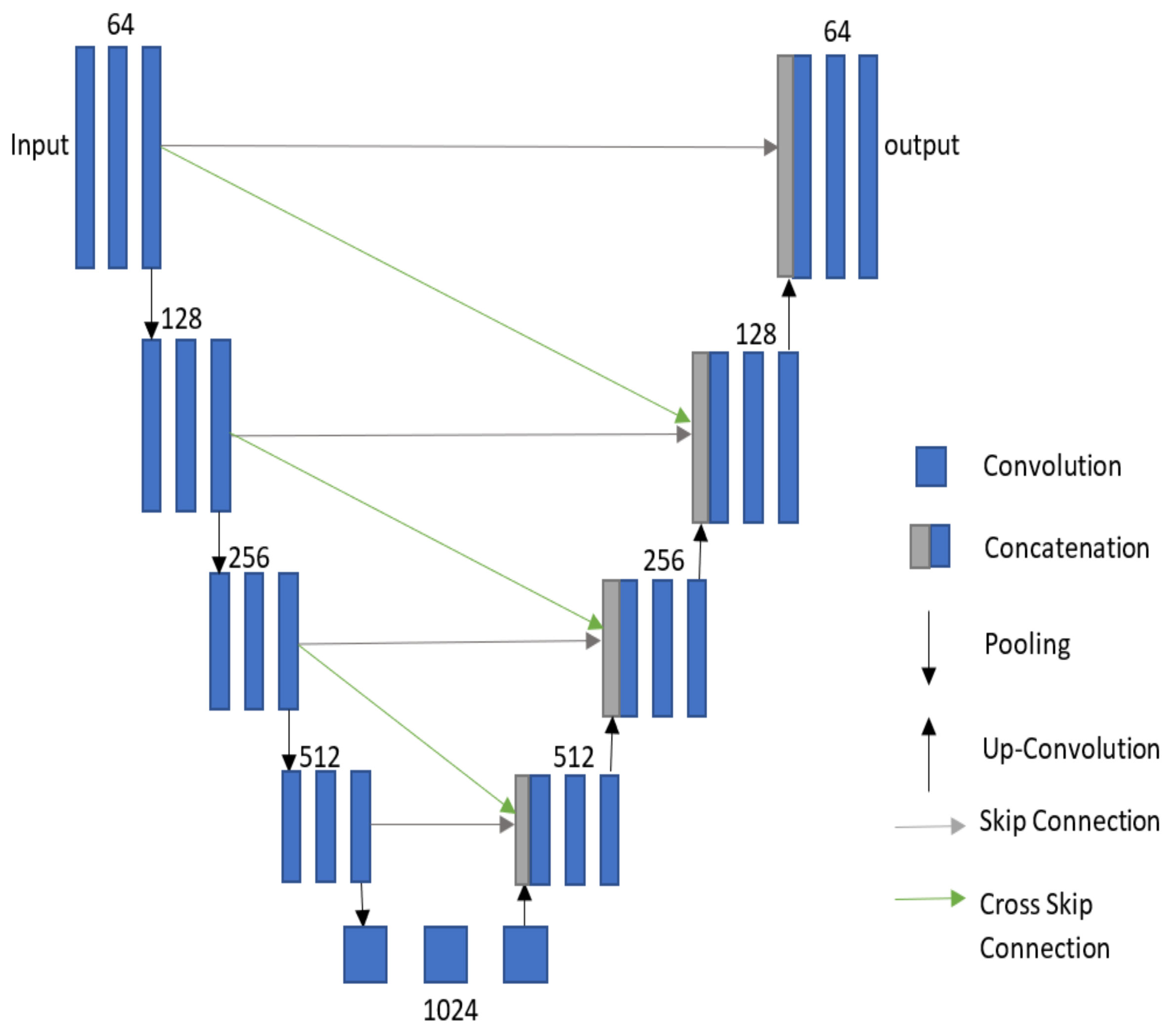


Fig. 38. UNET UC architecture.

The UNET UC model was trained for 16 epoch and the following results shown in table 8 was obtained during evaluation.

Table 8. Evaluation results of UNET UC.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISSION	ACCURACY	AUC-ROC
UNET UC	0.7970	0.8232	0.7904	0.9790	0.8589	0.9527	0.9774

6.8. Discussion :

6.8.1. Comparison :

Two variants of UNET with cross skip connections were built and evaluated, namely UNET C and UNET UC. The UNET C had skip connections from the lower level convolution block of the encoder to the higher level convolution block of the decoder, and the UNET UC had Cross skip connections from the upper-level convolution block of the encoder to the lower level convolution block of the decoder. The evaluated results of the two models are shown in table 9 below for comparison.

Table 9. Evaluation results of UNET C and UNET UC.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISION	ACCURACY	AUC-ROC
UNET C	0.7929	0.8174	0.7634	0.9831	0.8795	0.9525	0.9776
UNET UC	0.7970	0.8232	0.7904	0.9790	0.8589	0.9527	0.9774

From table 9 above, we can see that the UNET UC has the best MCC score and accuracy compared to UNET C. Figures 39 and 40 illustrate the AUC-ROC of UNET C and UNET UC. It is seen that the UNET UC has good potential for further improvisation and modifications.

X – Axis = False Positive Rate (FPR)

Y – Axis = True Positive Rate (TPR)

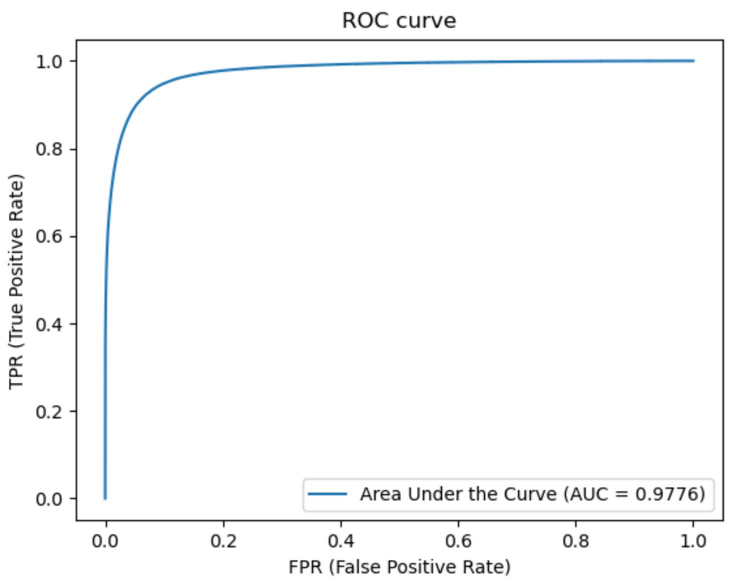


Fig. 39. AUC – ROC of UNET C.

X – Axis = False Positive Rate (FPR)

Y – Axis = True Positive Rate (TPR)

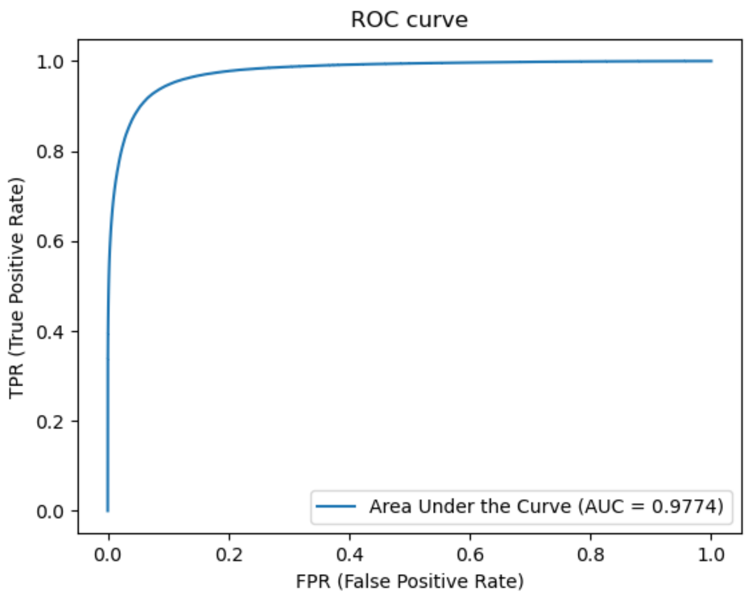
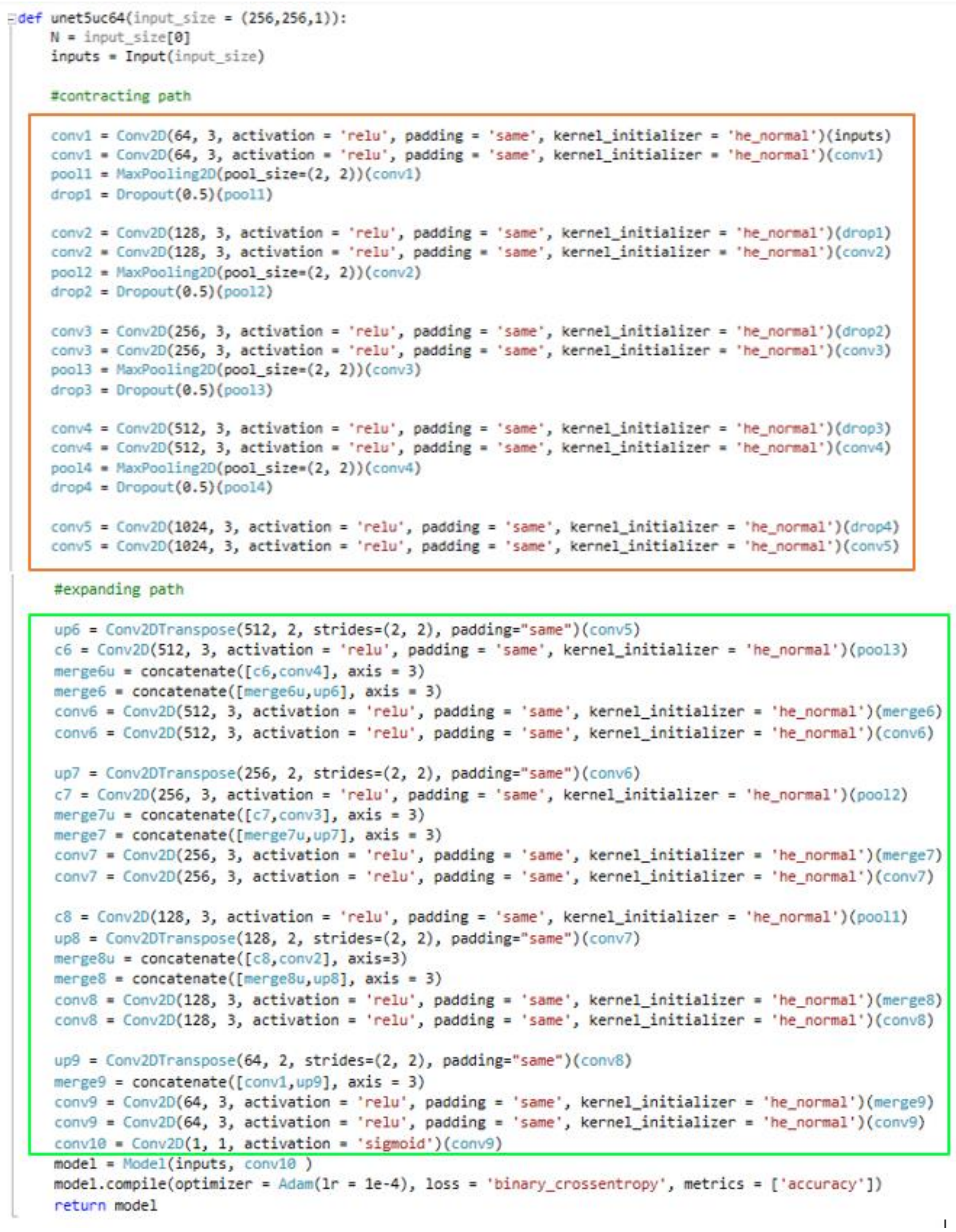


Fig. 40. AUC – ROC of UNET UC.

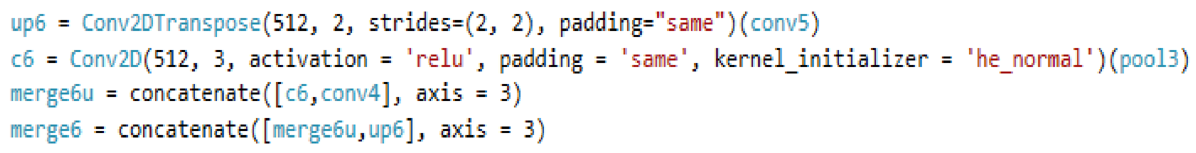
6.8.2. UNET UC Code :



```
def unet5uc64(input_size = (256,256,1)):
    N = input_size[0]
    inputs = Input(input_size)

    #contracting path

    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    drop1 = Dropout(0.5)(pool1)

    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    drop2 = Dropout(0.5)(pool2)

    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    drop3 = Dropout(0.5)(pool3)

    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)
    drop4 = Dropout(0.5)(pool4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop4)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv5)

    #expanding path

    up6 = Conv2DTranspose(512, 2, strides=(2, 2), padding="same")(conv5)
    c6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
    merge6u = concatenate([c6,conv4], axis = 3)
    merge6 = concatenate([merge6u,up6], axis = 3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2DTranspose(256, 2, strides=(2, 2), padding="same")(conv6)
    c7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
    merge7u = concatenate([c7,conv3], axis = 3)
    merge7 = concatenate([merge7u,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)

    c8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool1)
    up8 = Conv2DTranspose(128, 2, strides=(2, 2), padding="same")(conv7)
    merge8u = concatenate([c8,conv2], axis=3)
    merge8 = concatenate([merge8u,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv8)

    up9 = Conv2DTranspose(64, 2, strides=(2, 2), padding="same")(conv8)
    merge9 = concatenate([conv1,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)
    conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)
    model = Model(inputs, conv10 )
    model.compile(optimizer = Adam(lr = 1e-4), loss = 'binary_crossentropy', metrics = ['accuracy'])
    return model
```

Fig. 41. UNET UC code.

The code used for UNET UC is shown in figure 32. The contracting path of the UNET UC is highlighted by orange colour box, and the expanding path of the UNET UC is highlighted by green colour box. The contracting path consists of five convolutional blocks, including the base convolutional block represented as conv1, conv2, conv3, conv4 and conv5. The expanding path consists of four convolutional blocks represented as conv6, conv7, conv8, conv9. These convolutional blocks are the same as UNET 5 F 64 PS 64. However, in UNET UC, we have an extra skip connection known as cross skip connection. In this project, a simple convolutional layer is used to achieve this cross skip connection. This cross skip connection is first merged with the skip connection and then it is merged with the upsampled data. This process in the code is as shown in figure 42 below.



```
up6 = Conv2DTranspose(512, 2, strides=(2, 2), padding="same")(conv5)
c6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
merge6u = concatenate([c6,conv4], axis = 3)
merge6 = concatenate([merge6u,up6], axis = 3)
```

Fig. 42. Cross skip connection and merging process code.

The “up“ variable defines the process of upsampling and “c“ variable defines the cross skip connection. As we can see that a simple convolutional layer is used the achieve this cross skip connection. The cross skip connection is merged with the skip connection and then with the upsampling using the help of the concatenation function, as seen in figure 42.

6.8.3. UNET UC Training Visualisation:

The training accuracy of the UNET UC is visualised in figure 43. The pink colour line denotes the training accuracy, and the green colour line represents the validation accuracy. From figure 43, we can see that the training accuracy gradually increases as the number of epochs increases and the validation accuracy increases in the first few epochs and gradually reduces and becomes constant in the later epochs; this shows that the model is not losing data and learning much better compared to basic UNET.

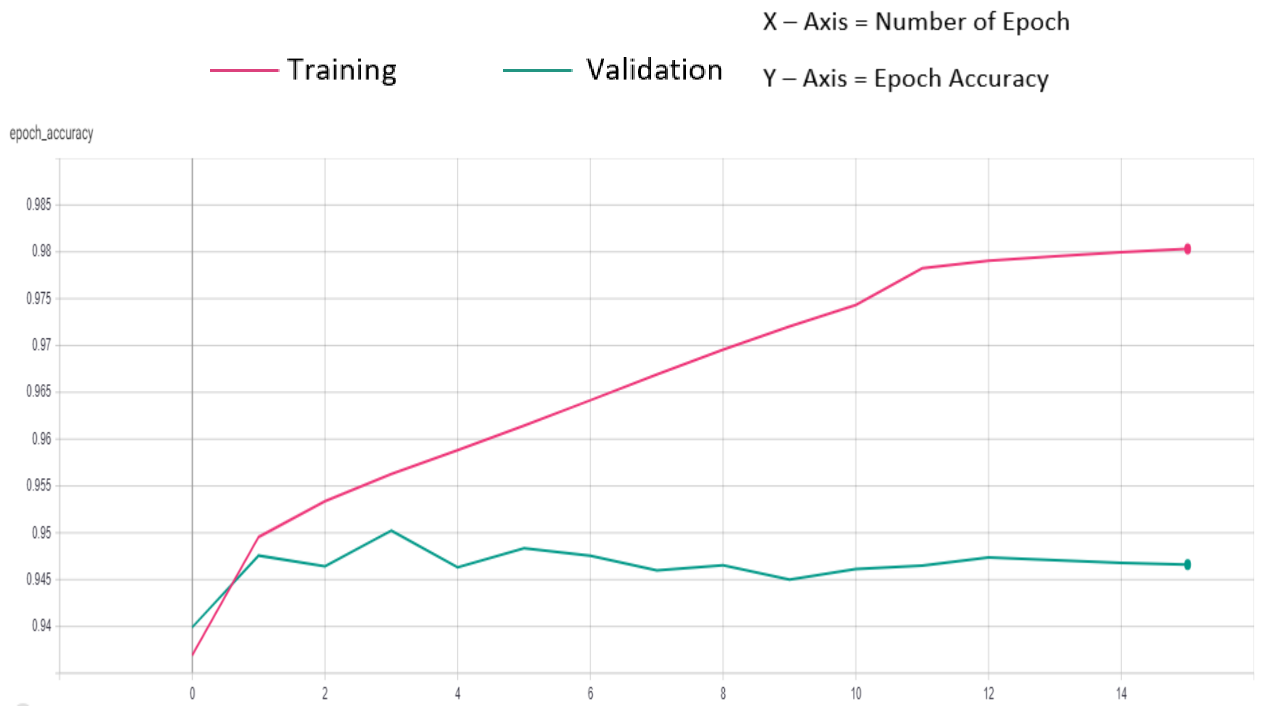


Fig. 43. Training Epoch Accuracy of UNET UC.

6.8.4. Segmentation Results :

The RBVS results of the UNET UC is shown in figure 44. Figure 44(a) shows the greyscale image, and the ground truth is shown in figure 44(b). The RBVS generated by UNET UC is shown in figure 44(c). The difference between the ground truth and the segmentation is shown in figure 44(d). In the difference image, the difference is marked in red colour. Few of the difference is due to the human error in ground truth failing to mark very thin blood vessels, few of the difference is due to failure of the UNET UC to segment thin blood vessels correctly and a majority of the difference is due to the very small difference in the size and of the line used in the ground truth and size of the line in the segmentations generated by UNET UC. As UNET UC segmentations are computer-generated, its segmentation lines are exactly the size of the retinal blood vessels. However, the ground truth is human traced segmentation of retinal blood vessels, and the size may differ compared to the actual size.

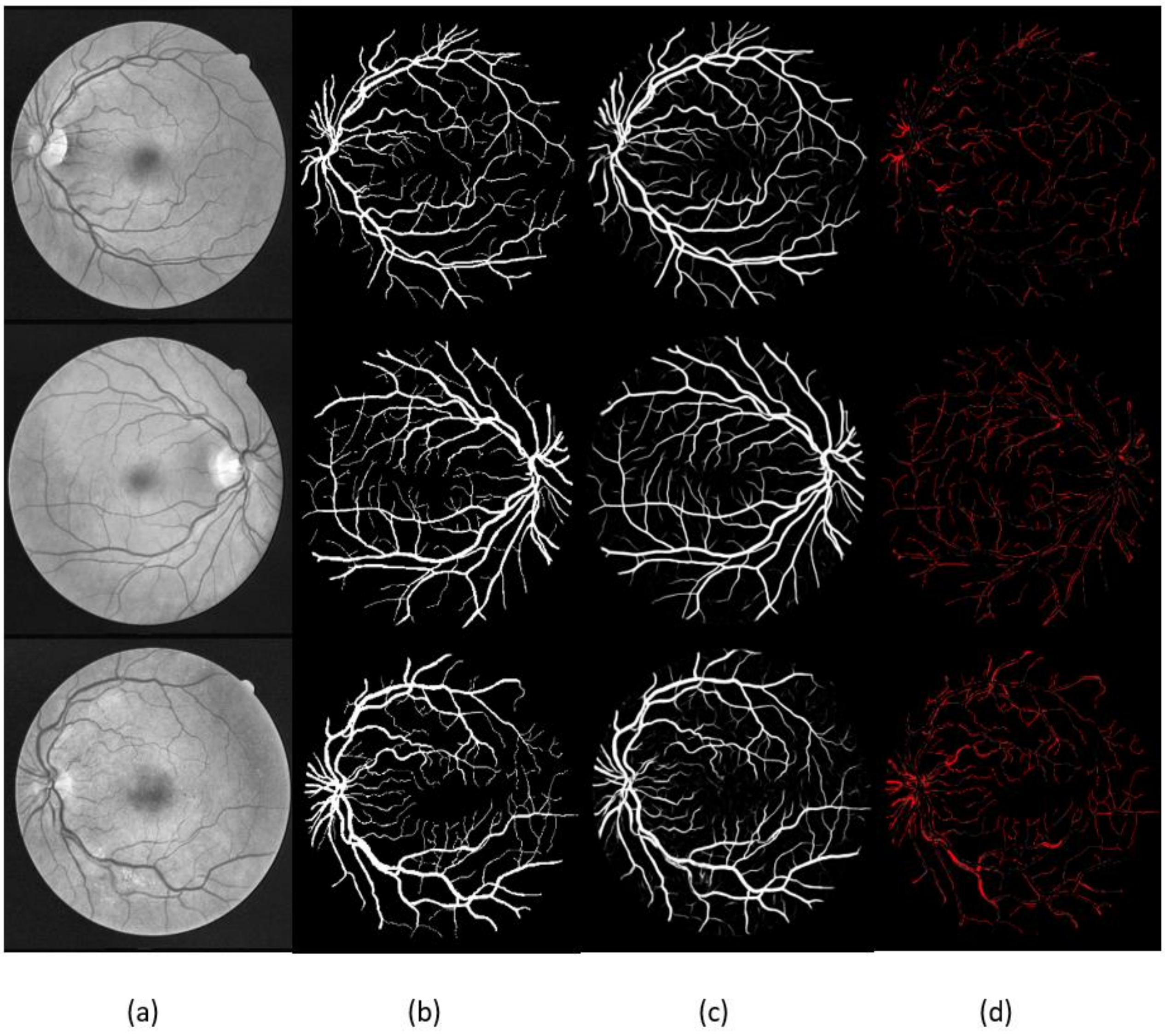


Fig. 44. Segmentation results of UNET UC.(a) Greyscale image, (b) Ground Truth, (c) UNET UC Segmentation, (d)Difference image.

6.9. UNET UCDA :

The architecture of UNET UCDA is represented in figure 45. The UNET UCDA consist of architecture similar to UNET UC. However, the difference is that UNET UCAD consists of a Dense convolutional network (figure 46) and Autrous Spatial pooling (figure 47) at the base of the network, as shown in figure 43.

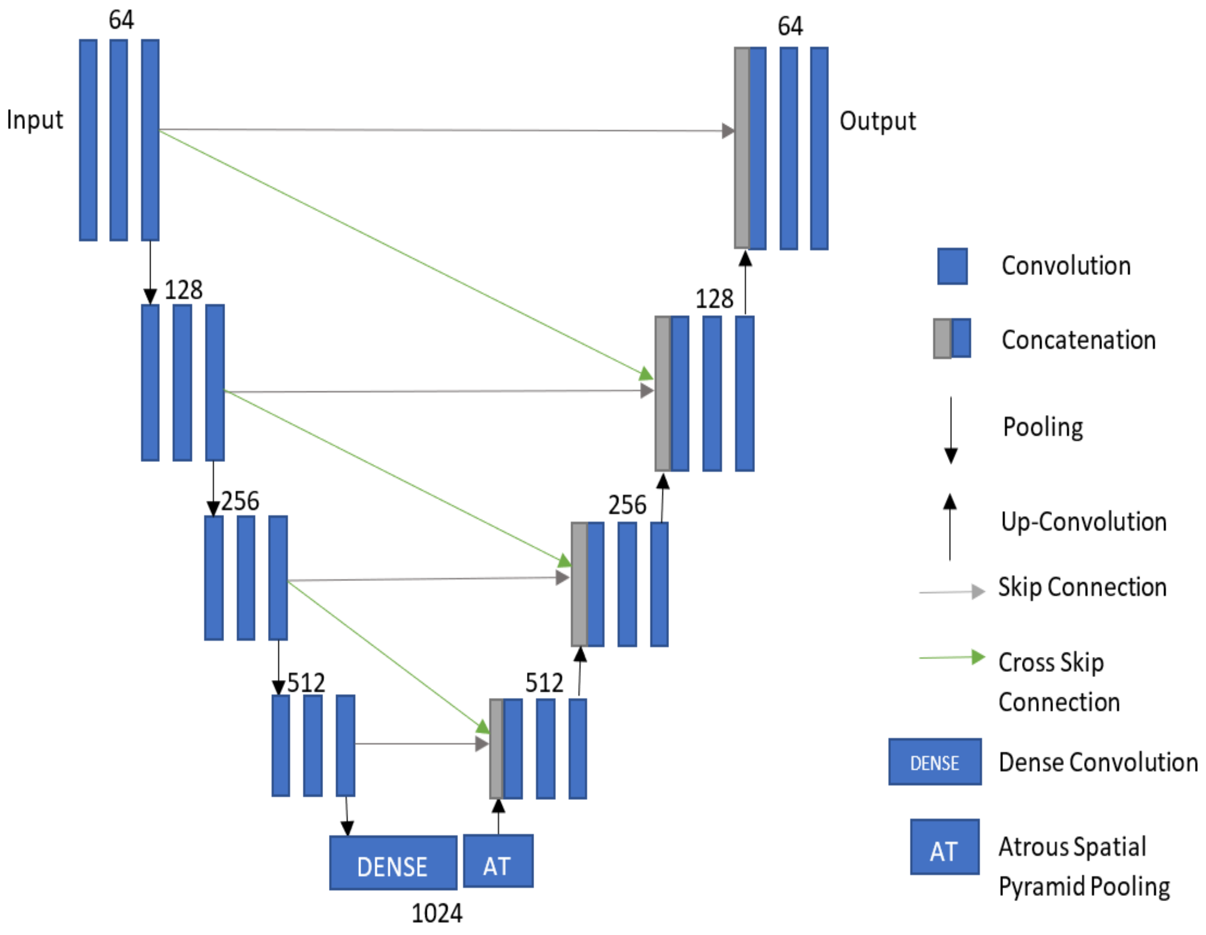


Fig. 45. UNET UCDA architecture.

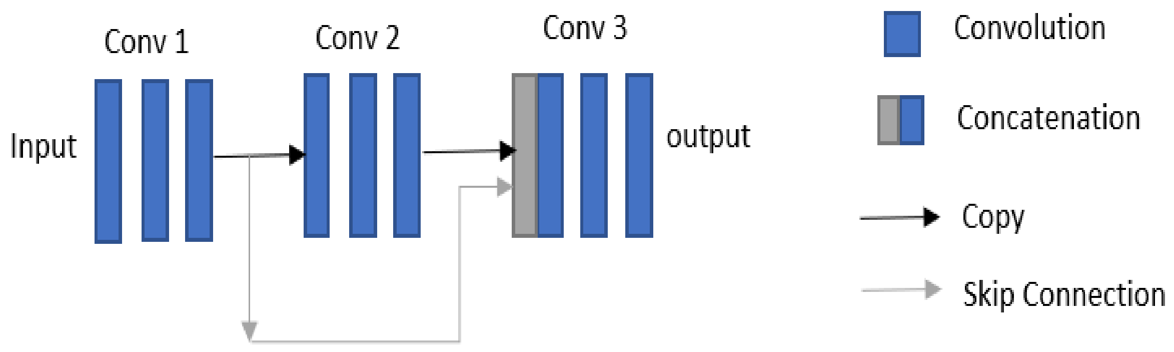


Fig. 46. Dense Network.

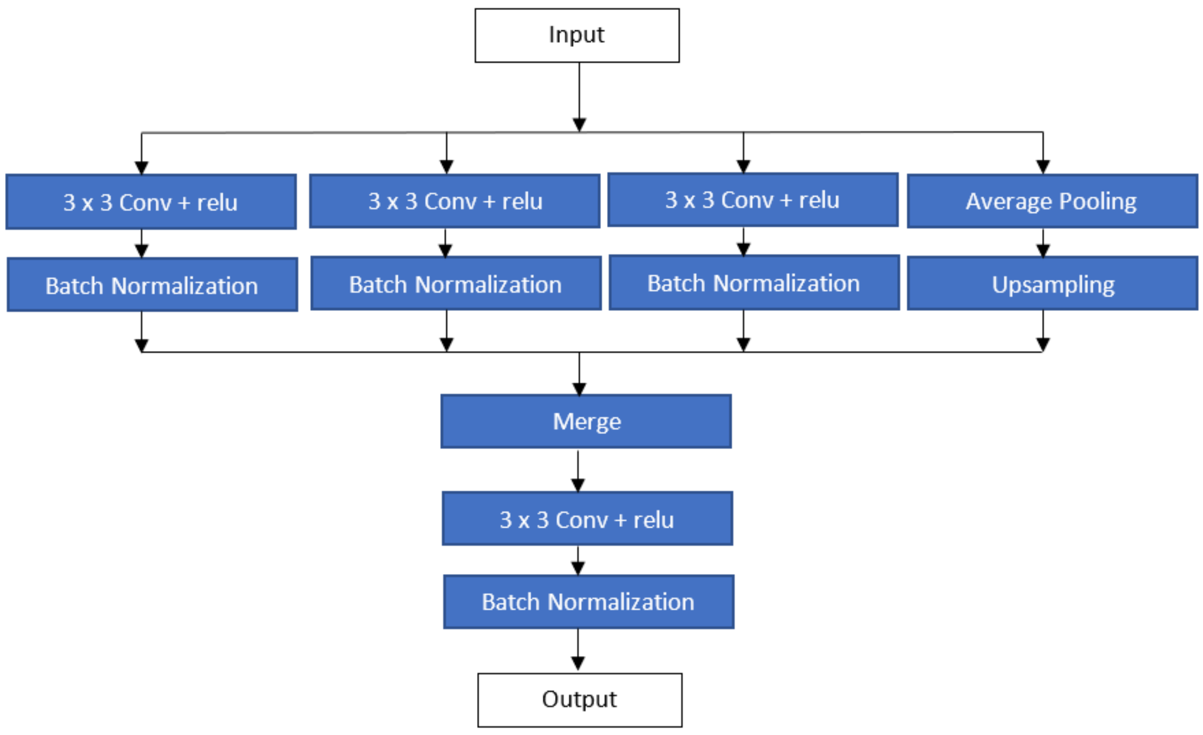


Fig. 47. Atrous Spatial Pyramid pooling.

The Atrous spatial pyramid pooling (ASPP) structure used in this experiment is as shown in figure 47. The input given to the ASPP is passed through three different convolutional layers with dilation rates 1, 2, 4 and an averaging pooling layer. The outputs of the convolutional layers are then given to batch normalisation, which keeps the mean close to zero and standard deviation close to 1. The output of average pooling is upsampled and then merged with the batch normalised outputs. After merging, the output is then passed through another final layer of convolution and batch normalisation.

The UNET UCDA model was trained for 16 epoch, and the following results shown in table 10 was obtained during evaluation.

Table 10. Evaluation results of UNET UCDA.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISION	ACCURACY	AUC-ROC
UNET UCDA	0.7968	0.8216	0.7744	0.9821	0.8751	0.9532	0.9774

With the introduction of dense block and atrous spatial pooling accuracy of the model has slightly increased. However, there are no improvements in MCC and F1 compared to UNET with cross skip connections.

6.10. UNET G UCDA :

The architecture of UNET G UCDA is represented in figure 48. The UNET G UCDA was inspired by the Gradient UNET (GNET)[12]. In the encoder part, the UNET G UCDA starts with convolutional layers with 64 filters and instead of downsampling like seen in traditional UNET, it is upsampled first. Then from the convolutional layers with 32 filters, downsampling is followed to the next lower layer as in traditional UNET. The UNET G UCDA consists of a dense network (figure 46) followed by Autrous spatial pyramid pooling (figure 47) before the data is upsampled to the decoder. In the decoder, the data from the adjacent encoder is concatenated, as seen in UNET UC. At the last convolutional layer with 64 filters the data is downsampled from the convolutional layers with 32 filters and concatenated with parallel convolutiol layer from the encoder.

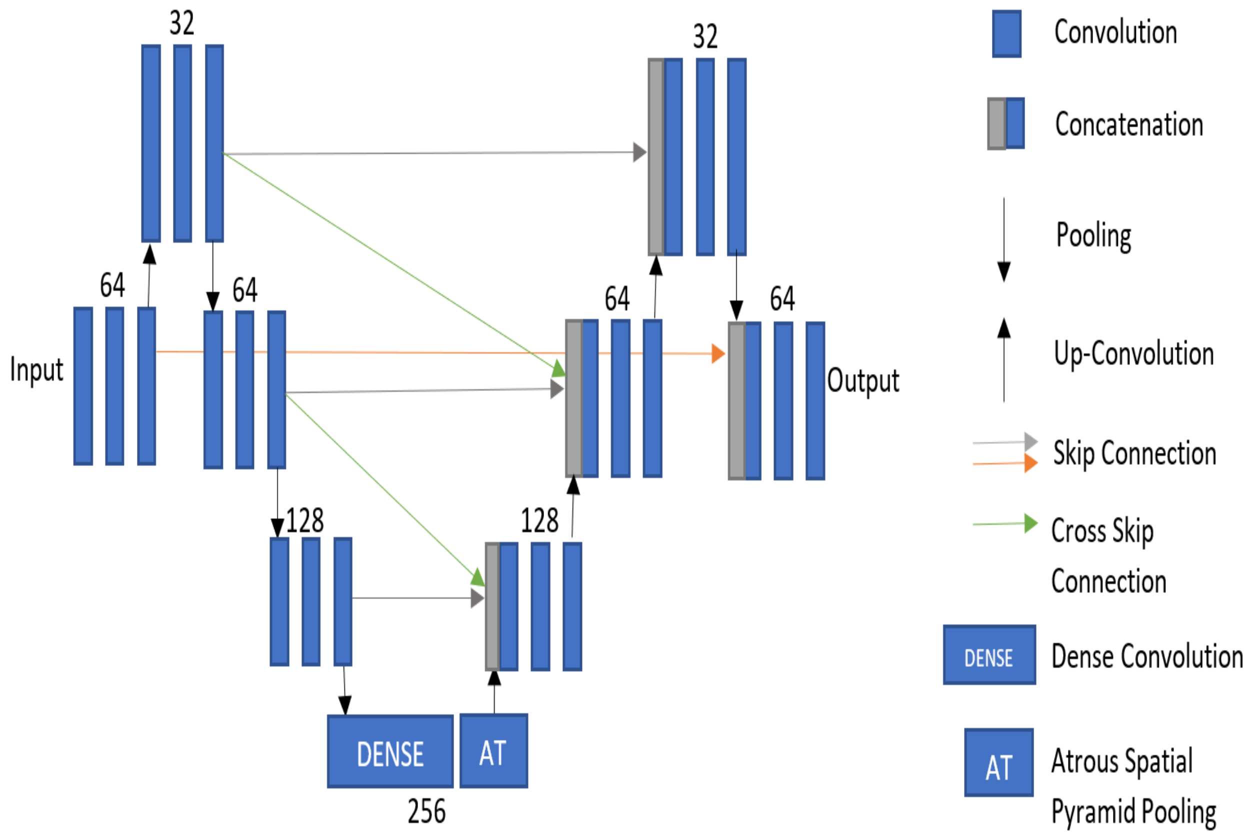


Fig. 48. UNET G UCDA architecture.

The UNET G UCDA model was trained for 16 epoch, and the following results were obtained during evaluation.

Table 11. Evaluation results of UNET G UCDA.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISION	ACCURACY	AUC-ROC
UNET G UCDA	0.8030	0.8296	0.8115	0.9765	0.8485	0.9536	0.9788

The UNET G UCDA has the best MCC score, Accuracy, and AUC-ROC compared to UNET UCDA, UNET C and UNET UC.

6.11. Discussion :

The UNET G UCDA produced efficient results thus far compared to other modified networks based on UNET. These results are compared and verified in the section below.

6.11.1. Comparison :

This section compares the evaluation results obtained by UNET G UCDA with a few of the best results obtained by other UNET models in this experiment. Table 12 below shows the tabulated results of the best UNET based models.

Table 12. Evaluation results of best UNET based models.

NETWORK	MCC	F1	SENSITIVITY	SPECIFICITY	PRECISSION	ACCURACY	AUC-ROC
UNET 5 F 64 PS 64	0.8005	0.8272	0.8073	0.9766	0.8482	0.9531	0.9784
UNET UC	0.7970	0.8232	0.7904	0.9790	0.8589	0.9527	0.9774
UNET G UCDA	0.8030	0.8296	0.8115	0.9765	0.8485	0.9536	0.9788

Comparing the evaluation results from the above table shows that UNET G UCDA has the best MCC score. The UNET G UCDA also has greater Accuracy and AUC-ROC compared to other networks. Figure 49 illustrates the AUC-ROC of UNET G UCDA. From the above results, the UNET G UCDA provides better segmentation than any other type of UNET models.

X – Axis = False Positive Rate (FPR)

Y – Axis = True Positive Rate (TPR)

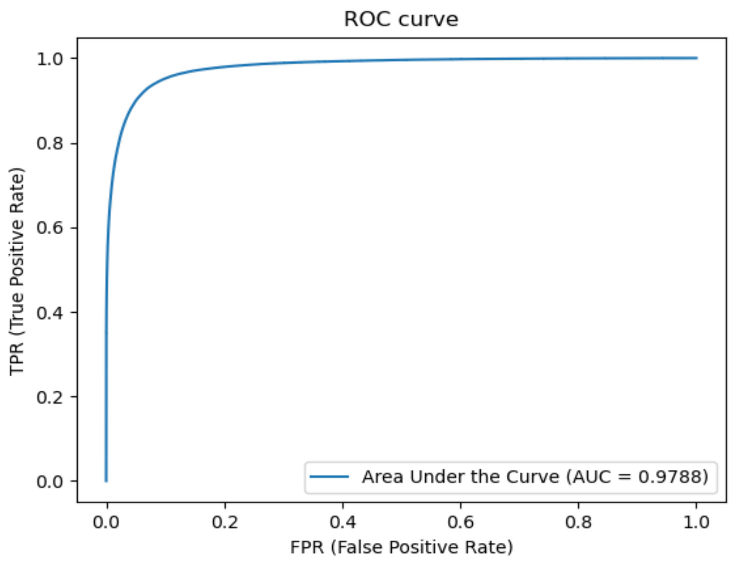


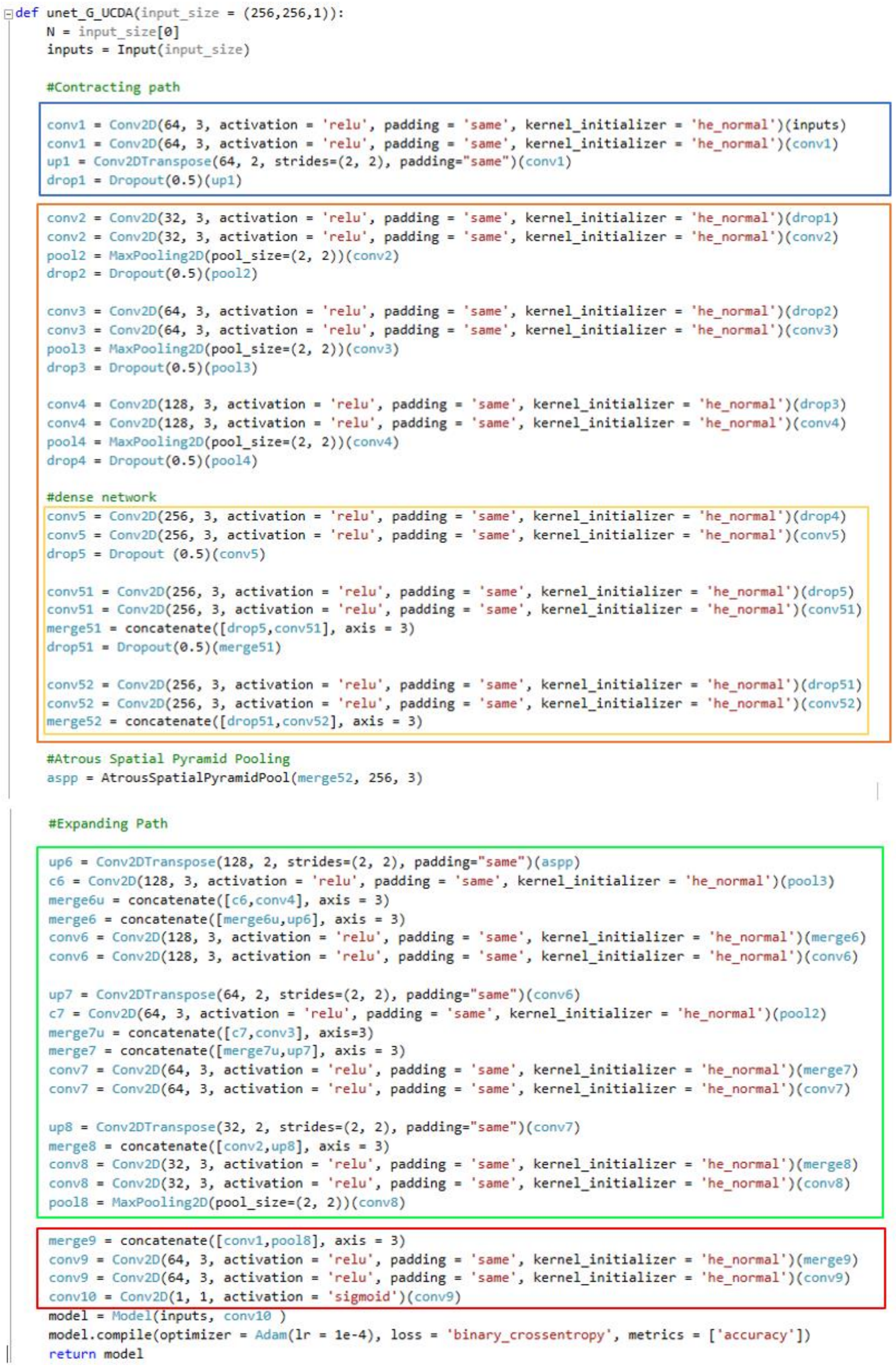
Fig. 49. AUC – ROC of UNET G UCDA.

Table 13. Network Parameters and Evaluation time.

NETWORK	Total Parameters	Trainable Parameters	Non-Trainable parameters	Evaluation Time
UNET 5 F 64 PS 64	31,03,593	31,03,593	0	3 minutes 29 seconds
UNET UC	35,676,353	35,676,353	0	3 minutes 8 seconds
UNET G UCDA	11,023,073	11,021,025	2,048	4 minutes 15 seconds

Table 13 shows the network parameters and evaluation time taken by each network to segment five retinal images. From the table, we can see that the UNET 5 F 64 PS 64 has the lowest number of parameters, and UNET UC has the highest number of parameters. In UNET G UCDA few paramters are non trainable. The evaluation time taken by each network to segment retinal images is subjected to the instance of GPU and memory available while processing. These evaluation time were achieved under the proposed experimental setup.

6.11.2. UNET G UCDA Code :



```
def unet_G_UCDA(input_size = (256,256,1)):
    N = input_size[0]
    inputs = Input(input_size)

    #Contracting path

    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
    up1 = Conv2DTranspose(64, 2, strides=(2, 2), padding="same")(conv1)
    drop1 = Dropout(0.5)(up1)

    conv2 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop1)
    conv2 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    drop2 = Dropout(0.5)(pool2)

    conv3 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop2)
    conv3 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    drop3 = Dropout(0.5)(pool3)

    conv4 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop3)
    conv4 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)
    drop4 = Dropout(0.5)(pool4)

    #dense network
    conv5 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop4)
    conv5 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv5)
    drop5 = Dropout (0.5)(conv5)

    conv51 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop5)
    conv51 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv51)
    merge51 = concatenate([drop5,conv51], axis = 3)
    drop51 = Dropout(0.5)(merge51)

    conv52 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(drop51)
    conv52 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv52)
    merge52 = concatenate([drop51,conv52], axis = 3)

    #Atrous Spatial Pyramid Pooling
    aspp = AtrousSpatialPyramidPool(merge52, 256, 3)

    #Expanding Path

    up6 = Conv2DTranspose(128, 2, strides=(2, 2), padding="same")(aspp)
    c6 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
    merge6u = concatenate([c6,conv4], axis = 3)
    merge6 = concatenate([merge6u,up6], axis = 3)
    conv6 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2DTranspose(64, 2, strides=(2, 2), padding="same")(conv6)
    c7 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
    merge7u = concatenate([c7,conv3], axis=3)
    merge7 = concatenate([merge7u,up7], axis = 3)
    conv7 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)

    up8 = Conv2DTranspose(32, 2, strides=(2, 2), padding="same")(conv7)
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv8)
    pool8 = MaxPooling2D(pool_size=(2, 2))(conv8)

    merge9 = concatenate([conv1,pool8], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)
    conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)
    model = Model(inputs, conv10 )
    model.compile(optimizer = Adam(lr = 1e-4), loss = 'binary_crossentropy', metrics = ['accuracy'])
    return model
```

Fig. 50. UNET G UCDA Code.

The code used for UNET G UCDA is shown in figure 50. The data from first convolutional block highlighted in blue colour is upsampled using max pooling and then passed to the next block. The contracting path of the UNET G UCDA is highlighted by orange colour box. The contracting path consists of four convolutional blocks including the dense convolutional block represented as conv2, conv3, conv4 and conv5. The convolutional block five is a dense network highlighted in yellow colour box, as shown in figure 50. The dense network consists of three convolutional blocks represented as conv5, conv51 and conv52. Each block input receives the output of all the previous blocks in the dense network; this is achieved by concatenation. The data output of the dense network is then sent to Atrous spatial pyramid pooling and then passed to the expanding path. The expanding path of the UNET G UCDA is highlighted by green colour box. The expanding path consists of three convolutional blocks represented as conv6, conv7, conv8. The output of convolutional block eight is downsampled and then given to the last convolutional block represented as conv9. The conv9 block is highlighted in red colour box in figure 50. The convolutional blocks, cross skip connections and skip connections are the same as UNET UC.

6.11.3. UNET G UCDA Training Visualisation:

The training accuracy and training loss of the UNET G UCDA is visualised in figure 51. The orange colour line denotes the training accuracy, and the blue colour line represents the validation accuracy. From figure 51, it can be seen that the training accuracy increases gradually as the number of epochs increases and the validation accuracy increases in the first few epochs and becomes constant in the later epochs. This shows that there is very less loss during overfitting, and the model learns much better compared to other models.

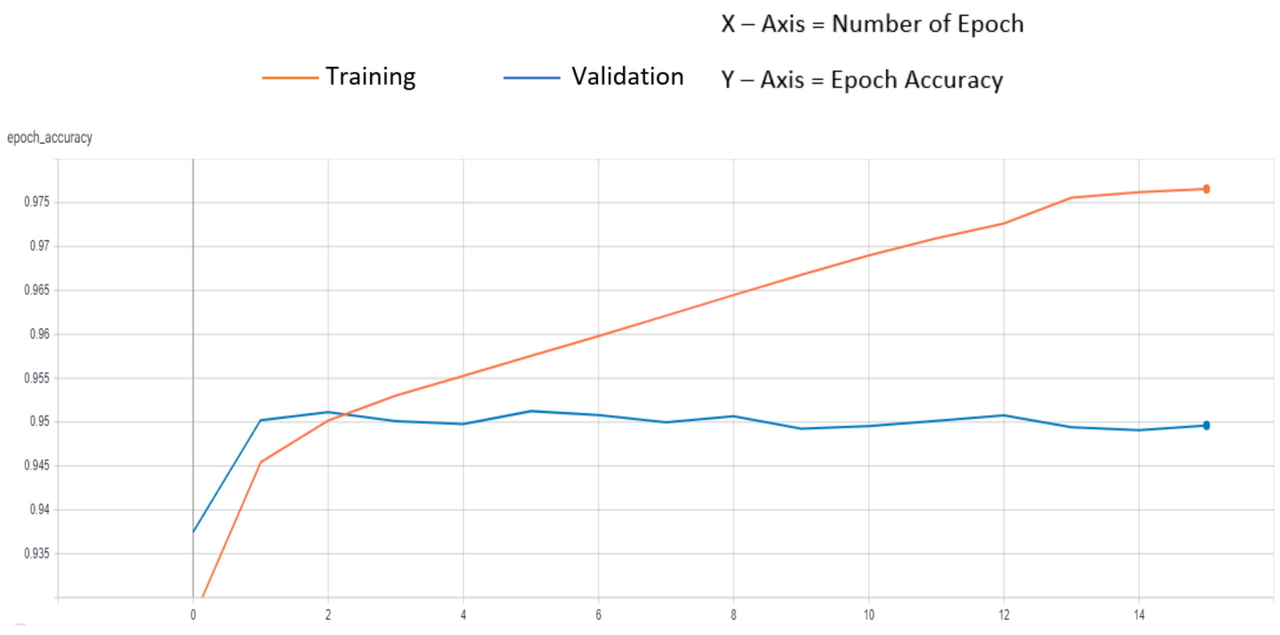


Fig. 51. Training epoch Accuracy of UNET G UCDA.

6.11.4. Segmentation results of UNET G UCDA:

The RBVS results of the UNET G UCDA is shown in figure 52. Figure 52(a) shows greyscale image, and the ground truth is shown in figure 52(b). The RBVS generated by UNET G UCDA is shown in figure 52(c). The difference between the ground truth and the segmentation is shown in figure 52(d).

In the difference image the difference is marked in red colour. Few of the difference is due to the human error in ground truth failing to mark very thin blood vessels, few of the difference is due to failure of the UNET G UCDA to segment thin blood vessels correctly and a majority of the difference is due to the very small difference in the size and of the line used in the ground truth and size of the line in the segmentations generated by UNET G UCDA. As UNET G UCDA segmentations are computer-generated, their segmentation lines are exactly the size of the retinal blood vessels. However, the ground truth is human traced segmentation of retinal blood vessels, and the size may differ compared to the actual size.

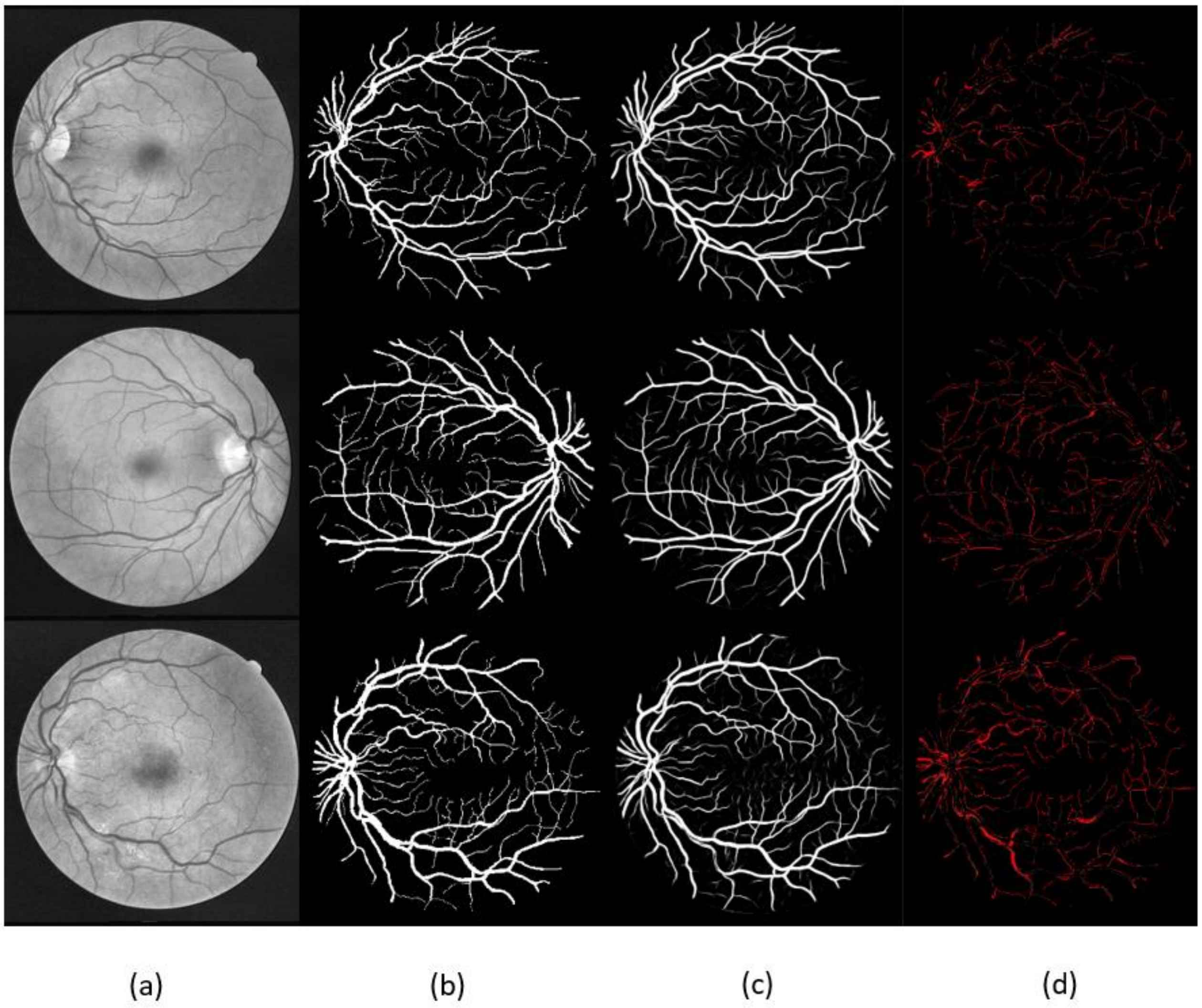


Fig. 52. Segmentation results by UNET G UCDA. (a) Greyscale image, (b) Ground truth, (c) UNET G UCDA Segmentation, (d)Difference in segmentation.

Conclusions

1. The research on RBVS by fully convolutional network was carried out. Different Convolutional networks were built based on the UNET to study the retinal vessel segmentation by convolutional networks.
2. By comparing all the different fully convolutional network models built based on UNET, we found that the UNET G UCDA provides better segmentation results.
3. .Compared to the basic UNET, the UNET G UCDA has improved results in terms of MCC, F1 Score, Sensitivity and Accuracy. The UNET G UCDA has an MCC score of 0.8030, F1 score of 0.8296, Sensitivity of 0.8115, Accuracy of 0.9536, and AUC-ROC of 0.9788, which are the best results provided by any UNET model under the proposed experimental setup.
4. By analysing the training data, UNET G UCDA generalises better than all the other UNET based models and provides the best retinal vessel segmentation based on MCC, F1 and Accuracy.
5. There is a scope for improvement in the performance of the proposed UNET G UCDA model by training it on larger datasets for retinal vessel segmentation.

List of references

1. ABRÀMOFF, MD, GARVIN, MK and SONKA, M. Retinal Imaging and Image Analysis. *IEEE Reviews in Biomedical Engineering*, 2010, vol. 3. pp. 169-208.
2. KANSKI, JJ and BOWLING, B. Kanski's Clinical Ophthalmology E-Book: A Systematic Approach. Elsevier Health Sciences, 2015.
3. THAM, Y., et al. Global Prevalence of Glaucoma and Projections of Glaucoma Burden through 2040: A Systematic Review and Meta-Analysis. *Ophthalmology*, 2014, vol. 121, no. 11. pp. 2081-2090.
4. FRAZ, MM, et al. Blood Vessel Segmentation Methodologies in Retinal Images–a Survey. *Computer Methods and Programs in Biomedicine*, 2012, vol. 108, no. 1. pp. 407-433.
5. SRINIDHI, CL, APARNA, P. and RAJAN, J. Recent Advancements in Retinal Vessel Segmentation. *Journal of Medical Systems*, 2017, vol. 41, no. 4. pp. 70.
6. ACHARYA, UR, et al. An Integrated Index for the Identification of Diabetic Retinopathy Stages using Texture Parameters. *Journal of Medical Systems*, 2012, vol. 36, no. 3. pp. 2011-2020.
7. XIAO, X., LIAN, S., LUO, Z. and LI, S. Weighted Res-Unet for High-Quality Retina Vessel Segmentation. IEEE, 2018.
8. GUO, C., et al. SD-Unet: A Structured Dropout U-Net for Retinal Vessel Segmentation. IEEE, 2019.
9. LECUN, Y., BOTTOU, L., BENGIO, Y. and HAFFNER, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998, vol. 86, no. 11. pp. 2278-2324.
10. LUO, Z., et al. Micro-Vessel Image Segmentation Based on the AD-UNet Model. *IEEE Access*, 2019, vol. 7. pp. 143402-143411.
11. HU, J., et al. S-Unet: A Bridge-Style U-Net Framework with a Saliency Mechanism for Retinal Vessel Segmentation. *IEEE Access*, 2019, vol. 7. pp. 174167-174177.
12. . XUE, L., et al. A Saliency and Gaussian Net Model for Retinal Vessel Segmentation. *Frontiers of Information Technology & Electronic Engineering*, 2019, vol. 20, no. 8. pp. 1075-1086.
13. . AKRAM, MU, KHALID, S. and KHAN, SA Identification and Classification of Microaneurysms for Early Detection of Diabetic Retinopathy. *Pattern Recognition*, 2013, vol. 46, no. 1. pp. 107-116.
14. AKRAM, MU and KHAN, SA Multilayered Thresholding-Based Blood Vessel Segmentation for Screening of Diabetic Retinopathy. *Engineering with Computers*, 2013, vol. 29, no. 2. pp. 165-173.
15. . ZEILER, MD. Adadelta: An Adaptive Learning Rate Method. *arXiv Preprint arXiv:1212.5701*, 2012.
16. RONNEBERGER, O., FISCHER, P. and BROX, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Springer, 2015.
17. ZHANG, P., et al. Urban Land use and Land Cover Classification using Novel Deep Learning Models Based on High Spatial Resolution Satellite Imagery. *Sensors*, 2018, vol. 18, no. 11. pp. 3717.
18. OLIVEIRA, A., PEREIRA, S. and SILVA, CA Retinal Vessel Segmentation Based on Fully Convolutional Neural Networks. *Expert Systems with Applications*, 2018, vol. 112. pp. 229-242.
19. HUANG, G., LIU, Z., VAN DER MAATEN, L. and WEINBERGER, K.Q. *Densely Connected Convolutional Networks.* , 2017.

20. AUGUSTAUSKAS, R. and LIPNICKAS, A. Improved Pixel-Level Pavement-Defect Segmentation using a Deep Autoencoder. *Sensors*, 2020, vol. 20, no. 9. pp. 2557.
21. ARTACHO, B. and SAVAKIS, A. Waterfall Atrous Spatial Pooling Architecture for Efficient Semantic Segmentation. *Sensors*, 2019, vol. 19, no. 24. pp. 5361.
22. COMBES, J., GROSSMANN, A. and TCHAMITCHIAN, P. Wavelets: Time-Frequency Methods and Phase Space Proceedings of the International Conference, Marseille, France, December 14–18, 1987. Springer Science & Business Media, 2012.
23. CHEN, L., et al. Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected Crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017, vol. 40, no. 4. pp. 834-848.
24. ZHANG, Z., ZHANG, K. and KHELIFI, A. Multivariate Time Series Analysis in Climate and Environmental Research. Springer, 2018.
25. GUPTA, N. Artificial Neural Network. *Network and Complex Systems*, 2013, vol. 3, no. 1. pp. 24-28.
26. O’SHEA, K. and NASH, R. An Introduction to Convolutional Neural Networks. *arXiv Preprint arXiv:1511.08458*, 2015.
27. SEKOU, T.B., HIDANE, M., OLIVIER, J. and CARDOT, H. From Patch to Image Segmentation using Fully Convolutional Networks--Application to Retinal Images. *arXiv Preprint arXiv:1904.03892*, 2019.
28. ZHOU, Z., SIDDIQUEE, MMR, TAJBAKHSH, N. and LIANG, J. Unet : Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. *IEEE Transactions on Medical Imaging*, 2019, vol. 39, no. 6. pp. 1856-1867.
29. JIANG, Y., ZHANG, H., TAN, N. and CHEN, L. Automatic Retinal Blood Vessel Segmentation Based on Fully Convolutional Neural Networks. *Symmetry*, 2019, vol. 11, no. 9. pp. 1112.
30. SERMANET, P., et al. Overfeat: Integrated Recognition, Localisation and Detection using Convolutional Networks. *arXiv Preprint arXiv:1312.6229*, 2013.
31. GIUSTI, A., et al. Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks. IEEE, 2013.
32. . HE, K., ZHANG, X., REN, S. and SUN, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015, vol. 37, no. 9. pp. 1904-1916.
33. FENG, Z., YANG, J. and YAO, L. *Patch-Based Fully Convolutional Neural Network with Skip Connections for Retinal Blood Vessel Segmentation.* IEEE, Sep 2017.
34. GUO, C., et al. SA-UNet: Spatial Attention U-Net for Retinal Vessel Segmentation, Apr 07, 2020. Available from: https://arxiv.org/abs/2004.03696.