ISSN 2063-5346

A NOVEL APPROACH OF DIABETIC RETINOPATHY IMAGE SEGMENTATION AND DETECTION USING DEEP LEARNING

Abirami A[1], Durga Naga Sai Shriya. R[2], Aishwarya. S[3], Jemimah Joyce .X[4]

Article History: Received: 01.02.2023 **Revised:** 07.03.2023 **Accepted:** 10.04.2023

Abstract

Diabetic retinopathy is a complex eye disease caused by diabetes. This causes damage to blood vessels and creates abnormal new vessels. These lead to vision loss at the end if not diagnosed properly and not treated. Our approach includes data collection, data pre-processing to remove Diabetic retinopathy images from collected data sets and standardize them. Feature extraction is done using segmentation architecture i.e., UNet to propose a segmentation strategy that separates structure and texture components and also improves the performance. Then DR fundus images are trained and then the trained DR images are classified as DR or No DR using DL neural network like Convolutional Neural Network (CNN).

Keywords: diabetic retinopathy, CNN, segmentation, proliferative, blood vessel, haemorrhage.

[1]Assistant Professor, Computer Science and Engineering, Easwari Engineering College, abirami.a@eec.srmrmp.edu.in
[2]Computer Science and Engineering, Easwari Engineering College, durganagasaishriyaragu@gmail.com
[3]Computer Science and Engineering, Easwari Engineering College, aishwaryasaran040101@gmail.com
[4]Computer Science and Engineering, Easwari Engineering College, jemimahxavier@gmail.com

DOI: 10.31838/ecb/2023.12.s1.052

I. INTRODUCTION

Diabetic retinopathy is a complex retinal eye disease which causes loss of vision that is irreversible. It is not usually observed by the patients in the beginning stages, as it does not have any clinical symptoms unless it has become advanced one. This occurs when sugar level reaches high and cause destruction to blood vessels like leakage in retinal parts and gets swollen or stops the blood flow. We have two

categories of diabetic retinopathy one is Proliferative and other one is non-proliferative where proliferative is considered to be a complex one and non-proliferative is considered to be the mild stage and doesn't show any symptoms. The novel, abnormal blood vessels develop in the retinal layer of proliferative type. The DL algorithms is generally used to detect DR. The process of DR detection using DL involves steps such as preprocessing of the image, feature extraction like haemorrhage and classification of DR. The first step is image preprocessing which involves various image improvisation methodologies and restoration techniques for the normalization of the retinal images. The existing models take into account the features like Standard deviation of the red, green and blue components, [1]blood vessel density of possible number of microaneurysms, actual number of microaneurysms, density of hard exudates. In this system the deployed CNN does not fix the position of the object and the orientation. It requires huge amount of data for model training so the retinal fundus images will be segmented to identify the exact affected area. The trained DL model is transformed into .h5 data file and deployed in angular framework in order to provide better UI and to predict the output as Severe DR / No DR. Features such as haemorrhage main blood vessels are taken into account for accurate recognition.

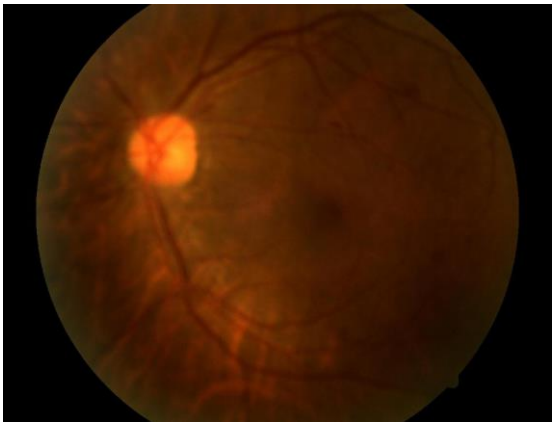

Fig 1. Severe Diabetic Retinopathy

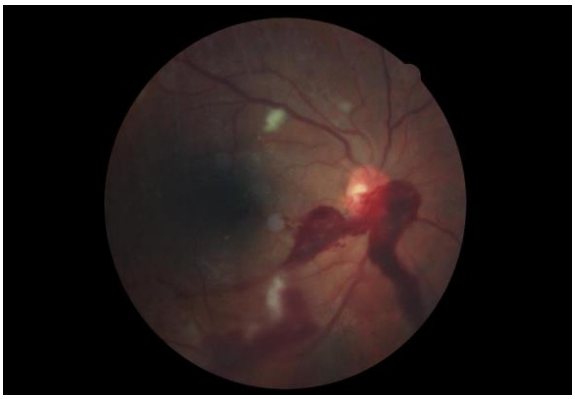

Fig 2. No Diabetic Retinopathy

II. MOTIVATION

- Advance recognition of the diabetic retinopathy disease provides chances for clinical practitioners to initiate the treatment and cure the disease at an earlier stage with high degree of accuracy.
- Prior detection is the key in diabetes mellitus as it avoids serious consequences.
- When a problem is found at the risk stage i.e., high blood sugar level in the eye, so the clinical surgeons and patients can take measures to prevent permanent damage to the heart, kidneys, eyes, nerves, blood vessels, and other vital organs.

III. RELATED WORKS

Detecting and classifying diabetic retinopathy (DR) is an important task in the field of ophthalmology and healthcare. There have been

many works related to DR detection and classification using various techniques like ML and DL. Here are some notable works and reviews:

[1] This systematic review explores the usage of ML algorithms for automated diagnosis of DR. The authors review 59 studies that use various machine learning techniques for DR detection and classification. The review highlights the advantages and limitations of each method and provides insights.

[2] This systematic review focuses on use of DL algorithms for DR analysing as well as classification. The authors review 29 studies that use DL techniques such as CNNs for the diagnosis of DR. The review also discusses the challenges and limitations of using DL for DR detection.

[3] This article provides overview for current state of art in automated DR detection using retinal fundus photographs. The authors review the various techniques and algorithms used for DR detection, including traditional ML and DL approaches. The review also discusses the challenges and limitations of automated DR detection in clinical practice.

[4] This article provides a summarized overview of DL techniques for the DR detection and classification. The authors review the various deep learning architectures used for DR diagnosis, including CNNs, RNNs, and GANs. The review also discusses the advantages and limitations of using DL for DR detection.

[5] This article provides an in-depth analysis of the use of DL for DR classification as well as detection. The authors review the various DL techniques used for the DR diagnosis, including CNNs, RNNs, and GANs. The review also discusses the challenges and limitations of using DL for the DR detection in real-world clinical settings.

[6] This paper proposes a DL based procedures for the automated detection as well as segmentation of diabetic retinopathy in fundus images. The proposed framework consists of CNN for the feature extraction, proceeded by a segmentation network that generates pixel-level segmentation maps. The authors evaluated their model on a publicly available dataset and achieved futuristic performance for both speed, accuracy.

[7] This paper presents DL based framework for automated segmentation of DR lesions in retinal images. The proposed framework consists of a modified U-Net architecture that incorporates dilated convolutional layers to capture context information. The authors evaluated their model on a large dataset and achieved futuristic performance for both speed, accuracy.

[8] This paper proposes a multi-stage deep learning approach for segmentation of DR lesions in fundus images. The proposed methodology includes two stages: the first stage performs coarse segmentation using a CNN-based model, while the second stage refines the segmentation using a conditional random field (CRF) model. The authors evaluated their approach on a large dataset and achieved futuristic performance for both speed, accuracy.

[9] This paper provides a DL based procedure for segmentation of diabetic retinopathy lesions using fundus photographs. The proposed model consists of a U-Net architecture that is trained using both supervised as well as unsupervised learning. The authors evaluated their approach on a publicly available dataset and achieved futuristic performance for both speed, accuracy.

[10] This paper proposes an automated segmentation approach for DR lesions from scanned laser ophthalmoscopy images. The proposed approach consists of a multi-stage segmentation framework that uses a CNN-based model for initial segmentation, followed by a refinement step using a graph cut algorithm. The authors evaluated their approach

on a large dataset and achieved futuristic performance for both accuracy and speed.

IV. PROPOSED PLAN

Our approach starts with the collection of datasets which includes retinal fundus images for training and testing. Now this dataset is analyzed for the stages like preprocessing and model training. The preprocessing is done to make the input images ready to fit into the Unet and CNN architectures .The preprocessed dataset is given to the Unet architecture for segmenting the required features like retinal blood vessels and for the generation of retinal mask images. Now segmented images will be trained using CNN architectures like ManualNet, AlexNet, LeNet for detecting diabetic retinopathy. The CNN architecture with the highest accuracy will be deployed into the application.

A. Data collection

The STARE dataset is available publicly for DR research. STARE stands for "Structured Analysis of the Retina".

The dataset includes images of both normal and abnormal retinas, including those with DR, and has been used to develop and evaluate automated DR detection algorithms.

STARE dataset is available for research purposes and can be accessed through the Ophthalmic Imaging Centre website or through various online repositories. It is important to note that using medical image datasets requires strict adherence to ethical guidelines and regulations.

We have collected retinal dataset specified as the following table.

Range	Specificity
0	No DR
1	Severe

This dataset consists of 200 normal, 200 DR images to train and test which will be considered for our analysis. This dataset helps to calculate the each and every value [2] of the features which is useful for detection.

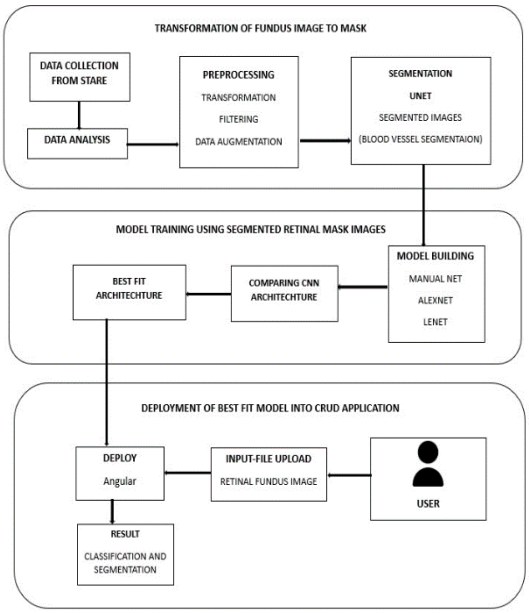

Fig 3. Proposed plan

B. Pre Processing

In order to determine the presence of the DR, the procedure in pre-processing stage includes rescaling, segmentation, horizontal flipping and ranging using shear range. Pre-processing is done to make sure the dataset is compatible, to display the suitable features, to remove noisy data. This step is very important. Then the images will undergo segmentation for differentiating normal and affected one.

With the help of Green Channel in the obtained fundus image i.e., RGB, the similarities and dissimilarities between features like the blood vessels, [3] haemorrhages can be seen. It is useful for analysis and classification purposes.

To enhance the features of the fundus images, it is divided into smaller blocks. The collected images would vary in terms of size and were cropped. In order to process these images, they had to be standardized by resizing or cropping it. First it should be cropped to a square. Using down sampling the new square image will be converted to the size 512 X 512 pixels. For some images back patch is included to make them uniform.

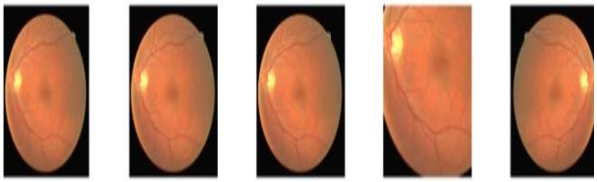

Fig 4. Fundus images

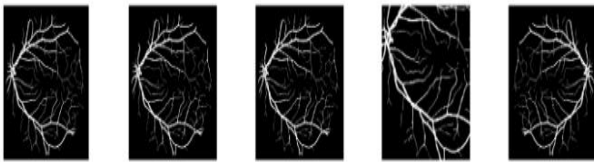

Fig 5. Mask images

C. Feature Extraction:

Features such as haemorrhage, main blood vessels are extracted using an architecture which is in the form of a 'U' called UNet. The UNet is one of the CNN architectures majorly used for segmentation of images and its levels are described below.

Encoder: The input retinal image is initially passed through a set of ConV and max pool layers in encoder path. These layers extract highly abstract as well as high level features from images to reduce its spatial resolution.

Decoder: The encoded features like haemorrhage are passed through a set of up sampling and ConV layers in the decoder path. These layers gradually increase the resoluteness of the feature maps and enable the model to produce a segmentation map which is of the same size equals to the DR image.

Skip connections: In order to conserve the resoluteness during segmentation, the UNet architecture includes skip connections that connect the encoder and decoder path at various levels. These connections pass the high-definition feature maps from the encoder directly to the corresponding layers in the decoder, [4] so that it helps the model to recover fine-grained details and produce accurate segmentations.

Output: The final layer of the decoder path produces a segmentation map that assigns label for each and every pixel in input retinal image. The output map can then be post-processed to remove small or noisy regions and improve accuracy of segmentation.

Training: The UNet model is trained using a set of labelled retinal images, where the ground truth segmentations are provided for each image. During training, the model learns to optimize the parameters of the encoder part and decoder part layers to produce accurate segmentations of the input images.

Inference: As soon as the UNet model is trained, it can be utilised to segment new retinal images. During inference, the model takes an input image and produces a segmentation map that assigns label to each and every pixel in the input retinal image. [4]The model can be trained using labelled retinal images and can be used to segment new images for diabetic retinopathy detection.

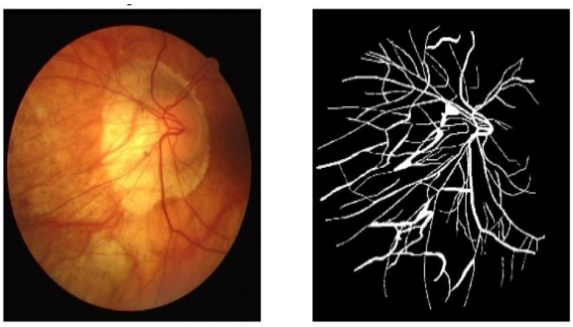

Fig.6 Fundus Image Mask Image

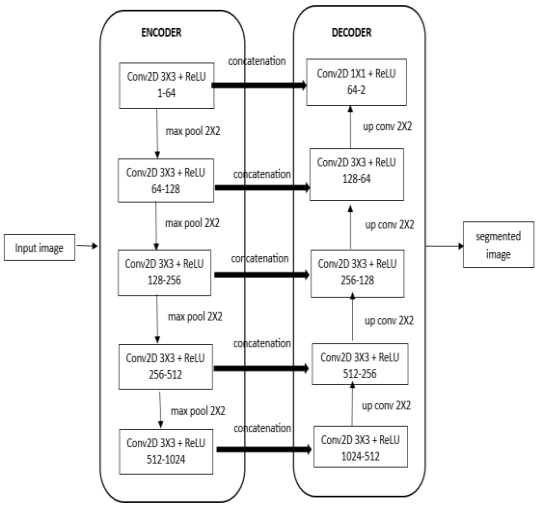

Fig 7. UNet architecture

D. Feature selection:

Diabetic retinopathy (DR) and its associated haemorrhages can be quantitatively evaluated using various metrics, including the number of haemorrhages and their size.

Here are a few formulas that are commonly used to calculate these metrics:

Number of haemorrhages:

The number of haemorrhages can be counted manually from retinal images or automatically using image processing algorithms. The formula to calculate the number of haemorrhages is simply the total count of haemorrhages present in the image.

$$count = x * y * \left(\frac{z}{2}\right)$$

where x -the greater diameter of the haemorrhage, y- 90 degree to x, z-total number of slices in haemorrhage CT to the considered thickness of slice.

Retinal haemorrhage:

The formula for calculating retinal haemorrhage in the eye depends on the type of haemorrhage being measured. Here are some common formulas used for different types of haemorrhages:

Dot and blot haemorrhages: The formula for calculating the number of dot and blot haemorrhages per disc area is:

$$H_{db} = \frac{N_h}{DA}$$

where,

H_{db} – Total number of dot and blot haemorrhages, N_h – Total number haemorrhage in disc area, DA – disc area

Flame-shaped haemorrhages: The formula for calculating the length of a flame-shaped haemorrhage is:

$$L_h = n * d$$

where L_h - Total length of the haemorrhage, n- Number of disc diameters the haemorrhage spans, d- Disc diameter.

Subretinal haemorrhages: The formula for calculating the volume of a subretinal haemorrhage is:

$$V_h = W_h * L_h * H_h * 0.52$$

where V_h −Volume of haemorrhage

W_h - Maximum width of haemorrhage L_h − Maximum length of haemorrhage H_h − Height of haemorrhage

TABLE I: HAEMORRHAGE COUNT ALGORITHM

Algorithm: Haemorrhage count
Input: Image - the retinal segmentation image Step 1: Set a threshold value Step 2: Convert the image to grayscale and apply pre-processing techniques Step 3: Identify the areas in the image that exceed the threshold value. Step 4: Apply dilation, erosion operations Step 5: Label the connected components in the image and count the label. Output: Count - the number of haemorrhages in the image.

E. Convolutional Neural Network

CNN is a kind of neural network which is suitable for the image classification. It includes multiple layers each of which carry out a specific operation on the input data.

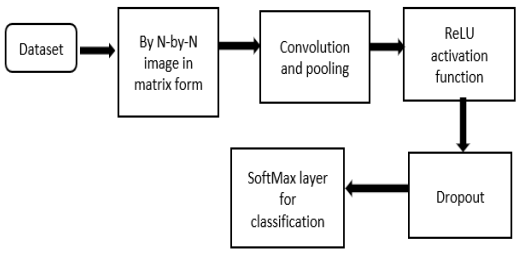

Fig 8. CNN work flow

The basic architecture of CNN typically consists of the layers as below:

Input layer: It takes in the original image datas, which is usually in the form of a 2D array values of the pixel.
Convolutional layer: This applies a series of filters (or) kernels to input images, to extract useful features such as edges, corners, and textures. Each filter produces a new "feature map" that highlights a specific pattern in the input retinal image [1].
In CNN, the feature map generated by applying filters to the input image is obtained through a mathematical operation called convolution. The convolution operation can be represented by the following formula:

$$(F * I)_{ab} = \sum_{K=-K}^{K} \sum_{l=-K}^{K} F_{k,l} \; I_{a-k,b-l}$$

F is the filter (or kernel), I is the input image, and K is the size of the filter (which is usually odd). The symbol * denotes the convolution operation, and (F * I) {a,b} represents the element of the feature map at position (a,b).

To extract multiple feature maps, we can use multiple filters, each of which produces a separate feature map. Let F^{(1)}, F^{(2)}, dots, F^{(n)} be n filters of size K times K. Then, the k-th feature map can be obtained as:

$$\left(F^k * I\right)_{a,b} = \sum_{p=-K}^{K} \sum_{q=-K}^{K} F_{p,q}^{k} \; I_{a-p,b-q}$$

This formula computes the k-th feature map by deform the k th filter with input images I. This operation is repeated for each filter to obtain multiple feature maps. The resulting feature maps can then be passed through activation functions and pooling layers for extracting suitable features from the input image.
The output matrix dimensions can be calculated as follows:
Assuming we have an input matrix of dimensions h, w, c, where h - height, w- width, and c-no. of channels, and we have a filter (or) kernel of dimensions fh, fw. where fh I- filter height and fw- filter width. We also have a padding value P and a stride value S.
The output matrix's dimension can be calculated using below formulae:

$$O_h = \left(\frac{(h+2P-fh)}{S}\right) + 1$$
$$O_w = \left(\frac{(w+2P-fh)}{S}\right) + 1$$
$$O_C = number\ of\ filters$$

where O_h - height of the output matrix, O_w - width of the output matrix, and O_C - no. of filters.

This formula takes into account the effect of padding and stride on the output dimensions. Padding adds extra rows and columns to the input matrix to ensure that the filter can be applied to the edges of the input matrix, while stride determines the spacing between each application of the filter.

Note that the output dimensions may not be integers, in which case we can round up or down as appropriate.
In case of using multiple filters on same image, convolution is carried out for each of them individually, the results stacks one on top of the other and are combined to single unit. The dimensions of the received tensor as follows.

$$Out_{dim} = \frac{(inp_{dim} + \ filter_{dim} + 2padding)}{stride} + 1$$

where:
inp_{dim} – dimensionality of input feature map
$filter_{dim}$ – spatial dimensionality of filter/kernel

"padding" - the no. of pixels added to the edges of the input feature map to pad it before the convolution operation. Padding is usually set to zero.
"stride" refers to the step size used to slide the filter/kernel over the feature map.

Each filter produces a separate feature map, and these feature maps are stacked together along the depth dimension to form the final output feature map.

ReLU layer:
This layer applies the Rectified Linear Unit (ReLU) activation function to the output of the convolutional layer. ReLU is a simple non-linear function that introduces non-linearity into the network and helps to improve its performance.

$$f(x) = max(0, x)$$

ReLU is utilized as the default activation function in neural network algorithms particularly CNNs as it is widely used.

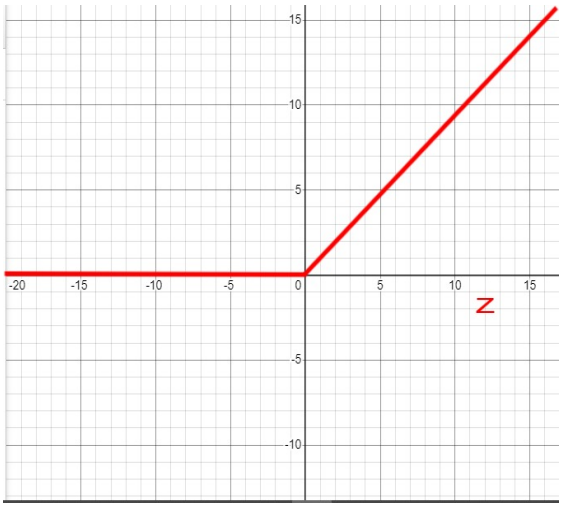

Fig 9. Graphical representation of ReLU

Pooling layer:
This layer downsamples output of previous layer by taking highest value in a small subregion of the feature map. This helps to reduce the spatial feature map size and makes as efficient network.
In a CNN, this layer is used for downsampling the feature maps produced by the convolutional layer. [5] The pooling layer works by splitting up the input feature map to the non overlapping areas, and applying a pooling operation to each region to produce a single output value. The formula for the most common type of pooling operation, max pooling, is as follows:

$$max\ pool(x_{a,b}) = max_{p=0}^{k-1}\ max_{q=0}^{k-1} x_{a+p,b+q}$$

where x_{a,b} is the element at position (a,b) in the feature map input, and K is size of pooling window . The formula computes the maximum value within a K times K window centered around the input element x_{a,b}. This maximum value is then used as the output value for the corresponding region in output feature map.
Alternatively, the average pool operation computes average value within each pooling region:

$$Avg\ pool_{(x_{a,b})} = \frac{1}{K^2} \sum_{p=0}^{K-1} \sum_{q=0}^{K-1} x_{a+p,b+q}$$

The formula computes the average value within a K times K window centered around the input element x_{a,b}. This average value is then used as the output value for the corresponding region in the output feature map.
Both max pooling and average pooling are commonly used in CNNs to lower the spatial dimensions of the feature maps while preserving the most important features.

Fully connected layer:
This layer connects each and every neuron in the preceding layer to the each and every neuron in present layer, and is used to combine the extracted features into a final prediction. This layer is often followed by an activation function called Softmax that converts output of the network to a probability distribution for the possible classes.
The softmax activation function is commonly used in the CNN's output layer for classification problems for more than one class. The softmax function takes a vector of random real time inputs and maps them to the

probability distribution over K classes. The formula for the softmax function is:

$$Softmax(X)_a = \frac{exp(x_a)}{\sum_{b=1}^{K} exp(x_b)}$$

The softmax function first exponentiates each element of the input vector f{x}, which results in a vector of non-negative values. The exponentiated values are then normalized through splitting them by the summation of all exponentiated values, which ensures that the resulting probabilities sum to 1. The denominator of the formula acts as a normalization factor to ensure that the outputs of the function form a valid probability distribution.

The softmax function is often utilized in the final layer of a CNN to output the predicted probabilities for individual class. The class which has the largest predicted probability will be considered as predicted class label for input.

Dropout layer:
A dropout layer is a normalization procedure in NN, including CNN, which prevents overfitting. In a CNN, a dropout layer randomly drops out a certain percentage of the neurons in the previous layer during training. This means that the selected neurons will not contribute to the forward pass or backpropagation during that particular training iteration [6].
The dropout layer in a CNN can be added after the ConV layer or after the FC layer. The rate of the dropout is hyper parameter that determines percentage of neurons to be dropped out during training. Typical values range from 0.1 to 0.5, depending on complexity of network and size of dataset.

The dropout layer helps prevent overfitting by forcing the network to learn more robust features. By randomly dropping out neurons during training, the network cannot rely too heavily on any single neuron or set of neurons. This encourages the network to learn multiple representations of the same data, making it more generalizable to unseen examples. During inference, the dropout layer is typically turned off or reduced to a lower value, allowing all neurons to contribute to the forward pass. The formula for the dropout layer can be expressed mathematically as follows:

Let x be the input tensor to the dropout layer, and p be the dropout rate (i.e., the fraction of input elements to drop out). The output y of the dropout layer is given by:

$$y = \frac{mask * x}{(1-p)}$$

where mask is a binary tensor of the same shape as x, with each element having a value of either 0 or 1. The elements of mask are randomly set to 1 or 0 with a probability of p and 1-p, respectively. In other words, mask is a random mask that determines which elements of x are dropped out during training.

During inference (i.e., when the model is used to make predictions), the dropout layer is typically turned off, and the output y is simply equal to the input x:

$$y = x$$

This ensures that the full power of the network is used to make predictions, without any element dropout or scaling.

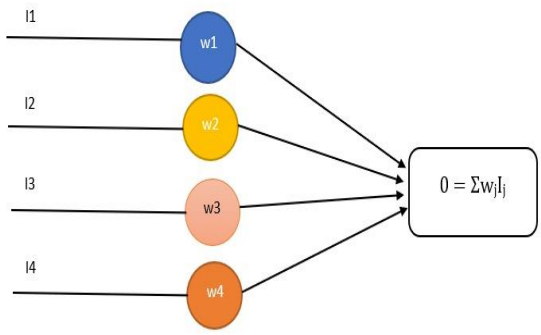

Fig 10. Dropout layer

Output layer:
This produces network's final output, i.e., a predicted class label or a set of probabilities for each possible class.
Overall, the architecture of CNN is designed to take out useful features from input data in a starting from low-level features like edges, gradually moving towards higher-level features in hierarchical manner, that are more representative of the underlying objects or scenes in the image.
The output of CNN is typically the predicted probability distribution over the target classes for a given input. The output layer of a CNN typically consists of a set of neurons or nodes, each corresponding to one of the target classes. The output values of these nodes can be calculated using the below formula:

$$Y = Softmax(Z)$$

where mathbf{z} is the input to the output layer, and mathbf{y}is the resulting vector of output values. [6]This ensures that the output values form a valid probability distribution over the target classes.

F. ManualNet:
This architecture is a simple and relatively shallow neural network that can be trained on image classification tasks. Convolutional layers and max pool layers help to extricate important features from fundus image, while the fully connected layers and softmax layer help to map these features to the desired output class probabilities.

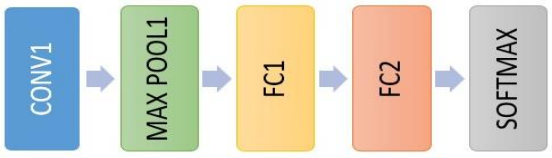

Fig 11. Representation of ManualNet

Convolutional layer:
This layer applies a combination of filters which are learnable to retinal input image inorder to extricate features. The output will be set of feature maps that capture different aspects of the input image.

Max pooling layer:
This layer downsamples the feature maps by selecting the maximum value in each non-overlapping subregion of the feature map. This reduces feature maps size and makes the model robust to small translations of input image.

Fully connected layer 1: This layer takes previous layer flattened output and applies a series of learnable weights which produce a new set of features.

Fully connected layer 2: This takes previous layer output and adds a set of learnable weights to produce the final set of features.

Softmax layer : This layer takes the output of previous layer, then applies the function called softmax to compute probability of each class.

G. Alexnet
The AlexNet architecture consists of 8 layers, including 5 ConV layers along with 3 FC layers. The ConV layers are followed by pooling layers which lower the spatial dimensions of output image. The final layer is a softmax layer, which provides the probability distribution over the output classes. The ReLU function is used in this layer. One of the key contributions of AlexNet is the use of larger filter sizes (11x11 and 5x5), which allow the network to capture more complex patterns in the input image. [6]The output of feature map is of 55 x 55 x 96.

Another contribution is the use of local response normalization, which helps to increase the generalization performance of the network. AlexNet also uses overlapping pooling, which allows the network to capture more spatial information. AlexNet uses random cropping, flipping to increase the training set size and prevents overfitting. Output size of convolution layer was calculated by the following formula,

$$out_{size} = \frac{(size_{in} - filter_{size} + 2 * padding)}{stride + 1}$$

Where:
$size_{in}$: input image size
$filter_{size}$: convolutional filter/kernel size
padding: the number of zero-padding pixels added to the edges of the input image/feature map
stride: the stride or step size of the convolutional filter/kernel.

To begin with we have maxpool layer which is of 3x3 with stride 2 so the feature map size is reduced to half of the previous size i.e., 27 x 27 x 96. The 2nd convolutional layer having each of 5x5in size is used with stride 1, padding 2 and ReLU function is used. This results with the feature map output 27 x 27 x 256. [7] Followed by this average pooling is applied of size 3x3 with the stride 2, this reduces the feature map size to its half of the precedence one i.e., 13 x 13 x 384.The model outputs the final convolution layer 256 filters with size 3x3 and results the feature map of size 13x13x256. Followed by the convolutional layers, now our model will have 84 neurons in fully connected layer and resulting an output value of 84 using ReLU. The final layer will be output layer consisting of 10 neurons. The function called Softmax is used which helps to determine the probability of the image and gets predicted based on the higher value.
Here we trained 28,039,482 parameters of the given retinal image using the AlexNet architecture.

TABLE 2: ALEXNET ARCHITECTURE ALGORITHM

Algorithm: AlexNet architecture
Input: X - input image dataset, Y - true labels, W - initialized weights 1: function AlexNet(X, Y, W): 2: Z1 ← Convolution (X, W1) # 96 filters of size 11x11 3: A1 ← relu(Z1) 4: P1 ← MaxPooling(A1) # pool with size 3x3 5: Z2 ← Convolution (P1, W2) # 256 filters of size 5x5 6: A2 ← relu(Z2) 7: P2 ← MaxPooling(A2) # pool with size 3x3 8: Z3 ← Convolution (P2, W3) # 384 filters of size 3x3 9: A3 ← relu(Z3) 10: Z4 ← Convolution (A3, W4) # 384 filters of size 3x3 11: A4 ← relu(Z4) 12: Z5 ← Convolution (A4, W5) # 256 filters of size 3x3 13: A5 ← relu(Z5) 14: P3 ← MaxPooling(A5) # pool with size 3x3 15: F ← flatten(P3) 16: Z6 ← FullyConnected (F, W6) # 4096 neurons 17: A6 ← relu(Z6) 18: Z7 ← FullyConnected(A6, W7) # 4096 neurons 19: A7 ← relu(Z7) 20: Z8 ← FullyConnected(A7, W8) # number of classes 21: loss ← cross_entropy_loss(Z8, Y) 22: gradients ← compute_gradients(loss, W) # Compute gradients for each layer 23: return gradients, loss Output: score of AlexNet trained model on test dataset

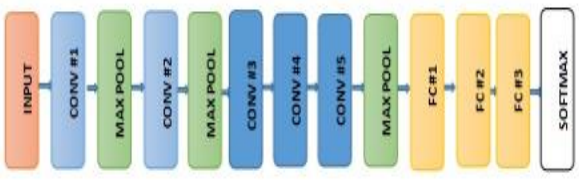

Fig 12. Representation of AlexNet

TABLE 2: ALEXNET ARCHITECTURE

Layers	**Filters**	**Size of the filter**	**Stride**	**Feature map size**
Input layer	-	-	-	227x227x3
Conv 1	96	11x11	4	55x55x96
Max Pool	-	3X3	2	27x27x96
Conv 2	256	5X5	1	27x27x256
Max Pool 2	-	3X3	2	13x13x256
Conv 3	384	3x3	1	13x13x384
Conv 4	384	3x3	1	13x13x256
Conv 5	256	3x3	1	13x13x256
Max pool3	-	3x3	2	6x6x256
Dropout 1	0.6	-	-	6x6x256

H. LeNet:

the LeNet model network has 5 layers with some parameters which are learnable one. This model consists of three groups of high convolutional layers associated with the Average Pooling operation. Followed by the convolution layer and by the average pooling it has 2 fully connected layer. Then the classifier called Softmax is used to classify the retinal images as no DR and severe DR. The input retinal image given to the model will be of 32x32x1 image which will be of grayscale image, so the count of the channels will be one. Then the first convolution performance is applied through 6 filters of size 5X5 filter. The activation function ReLU is used in this layer. [7]It results in the feature map 28x28x6 of size. The output size of the convolution layer was calculated by the following formula.

$$O_l = \frac{I - K + 2P}{S} + 1$$

Where - O_l: output size in the convolutional layer I: input size (width or height) of the previous layer K: kernel size

P: amount of padding applied to the input

S: stride.

Then the average pool is applied of 2x2 with stride 1 so the feature map size is reduced to half of the previous size i.e., 14x14x6. The 2nd convolutional layer having 16 filters each 5x5 in size is used with stride 1 and ReLU is used. This results with the feature map output 10x10x16. Followed by this average pooling is applied of size 2x2 with the stride 2, this reduces the feature map size to its half of the precedence one i.e. 5x5x16.

The model outputs the final convolution layer120 filters with size 5X5 and results the feature map of size 1X1X120 and gets flattened to 120. Followed by the convolutional layers, now our model will have 84 neurons in a fully connected layer and resulting an output value of 84 using ReLU. Final layer will be output layer consisting of 10 neurons since the images were classified into two distinct classes. The function called Softmax is used which helps to determine the probability [8]of the image and gets predicted based on the higher value. Here we have trained 6,461,186 parameters of the given retinal image using the LeNet architecture.

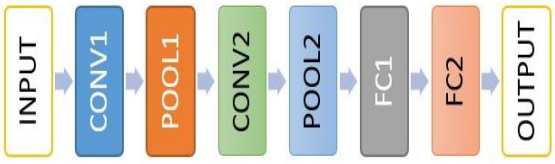

Fig 13 : LeNet architecture

TABLE 2: LENET ARCHITECTURE ALGORITHM

Algorithm: LeNet architecture
Input: X - input image dataset, Y - true labels, W - initialized weights 1: function LeNet (X, Y, W): 2: Z1 ← Convolution(X, W1) # 6 filters of size 5x5

3: A1 ← reLU(Z1)
4: P1 ← MaxPooling(A1) # pool with size 2x2
5: Z2 ← Convolution (P1, W2) # 16 filters of size 5x5
6: A2 ← reLU(Z2)
7: P2 ← MaxPooling(A2) # pool with size 2x2
8: F ← flatten(P2)
9: Z3 ← FullyConnected(F, W3) # 120 neurons
10: A3 ← reLU(Z3)
11: Z4 ← FullyConnected(A3, W4) # 84 neurons
12: A4 ← reLU(Z4)
13: Z5 ← FullyConnected(A4, W5) # number of classes
14: loss ← cross_entropy_loss(Z5, Y)
15: gradients ← compute_gradients(loss, W) # Compute gradients for each layer
16: return gradients, loss
Output: score of LeNet trained model on test dataset

TABLE 3: LENET ARCHITECTURE

Layers	**No of Filters**	**Size of the filter**	**Stride**	**Feature-map size**
Input layer	-	-	-	32x32x1
Conv2D 1	6	5x5	1	28x28x6
Avg. pooling layer 1	-	2x2	2	14x14x6
Conv2D 2	16	5x5	1	10x10x6
Avg. pooling layer2		2x2	2	5x5x6
Conv2D 3	120	5x5	1	120
Fully connected layer 1	-	-	-	84
Fully connected layer 2	-	-	-	10

V. PERFORMANCE EVALUATION

Performance evaluation for diabetic retinopathy segmentation and classification is typically carried out by comparing the output of an algorithm. The following are some measures used to estimate the performance of segmentation and classification algorithms:

Accuracy:

Accuracy is a commonly used metric for estimating performance of CNN model. It calculates the correctly classified samples percentage to the overall samples in the test set. The formula for calculating accuracy score in a binary classification problem (where there are only two classes: positive and negative) is:

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

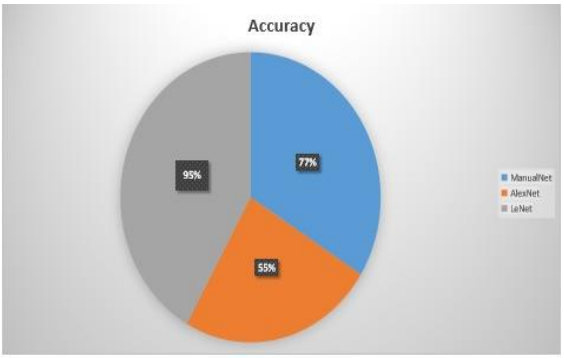

Fig 14. Accuracy representation for CNN architectures

TABLE 3: ACCUARCY REPRESENTATION

Architecture	**Accuracy**
AlexNet	0.54167
ManualNet	0.77381
LeNet	0.95312

VI.ACCURACY AND LOSS ANALYSIS:

The loss function calculates the difference between the predicted output and true output of the network. The goal is to lower the loss function. The lower the loss, the better the performance of the network. Accuracy is another important metric that measures the percentage of correctly classified samples in the dataset. It is calculated by dividing the no. of correct predictions by the total no. of predictions. The higher the accuracy, the better the performance of the network. However, accuracy alone may not be a sufficient evaluation metric, as it does not account for the distribution of the classes or the cost of misclassification errors. Both the loss and accuracy are used to monitor the performance of the CNN during the training process, and they are used to make decisions such as adjusting the learning rate, early stopping, or selecting the best model. The model accuracy and model loss for detection of diabetic retinopathy using AlexNet and LeNet is as follows

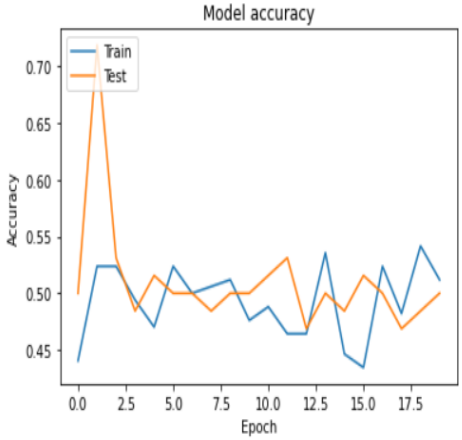

Fig 15. Accuracy for AlexNet

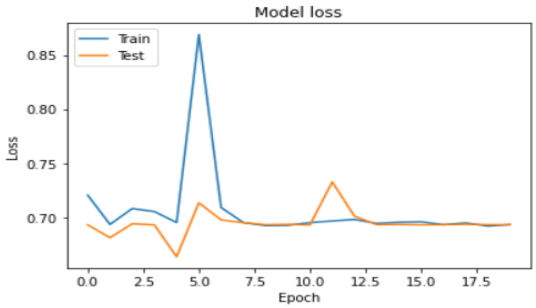

Fig 16. Loss for AlexNet

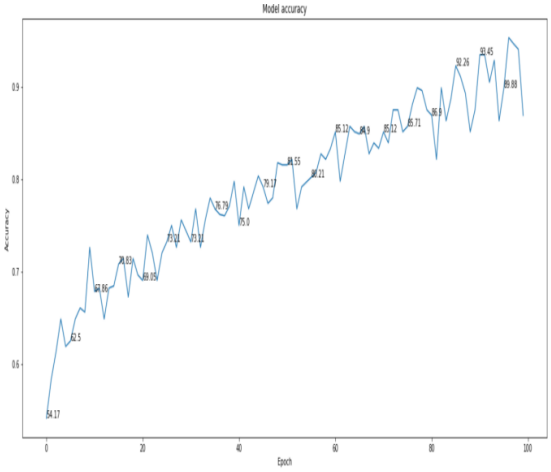

Fig 17. Accuracy for LeNet

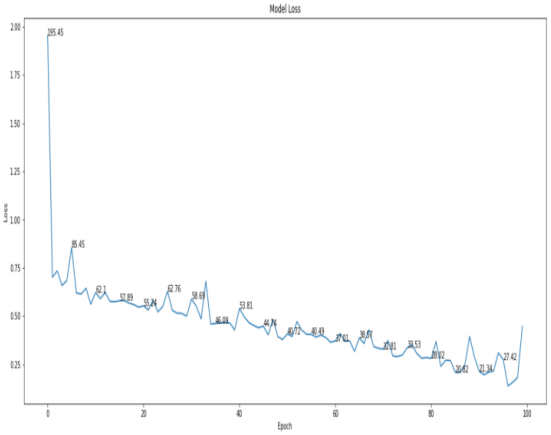

Fig 18. Loss for LeNet

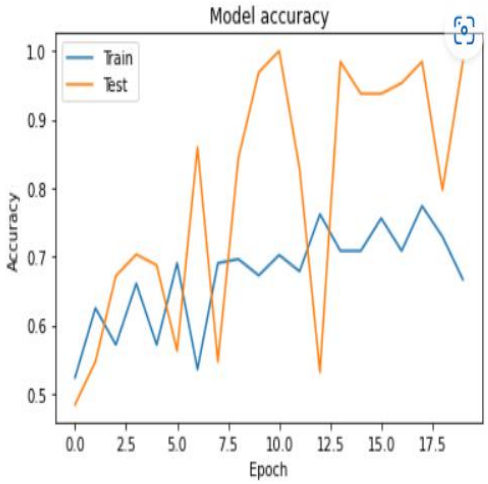

Fig 19. Accuracy for ManualNet

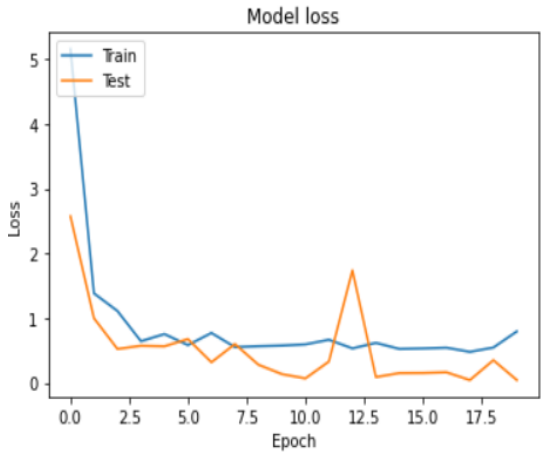

Fig 20. Loss for ManualNet

Conclusion and Future work:

The detection and segmentation of DR is a crucial task in the early diagnosis and treatment of the disease. With advancements in deep learning, automated systems for the detection and segmentation of DR has been developed. These systems have shown best results with high accuracy and efficiency. Automated DR detection and segmentation systems can provide Fig 18. Loss for LeNet a cost-effective and scalable solution. However, these systems still face some challenges, such as the need for large datasets, handling variability in image quality, and generalizability across different populations. Further research and development are required to address these challenges and improve the accuracy and efficiency of systems. [3]The detection and segmentation of DR is an active area of research, and there are several directions for future work. Here are some possible avenues for further research and development in this field:

Large and diverse datasets:
Future work can focus on collecting and curating more extensive and diverse datasets that can capture the variability in DR across different populations and disease stages.

Explainable AI:
Future work can focus on developing explainable AI methods that can provide insights into the decision-making process of deep learning models for DR detection and segmentation.

Transfer learning:
Future work can explore the use of transfer learning for DR detection and segmentation, where pre-trained models on related tasks or datasets can be used to improve the performance of models on new DR datasets.

Multimodal imaging:
Future work can focus on developing multimodal imaging systems that can integrate information from multiple imaging modalities to improve the accuracy and efficiency of DR detection and segmentation.

Real-time applications:
Future work can focus on developing real-time DR detection and segmentation systems that can run on low-power devices such as smartphones and portable cameras.

References

[1] M. Kumar, "Train Track Crack Classification Using CNN," *Trans Tech Publications*, p. 20, 2023.

[2] "Artificial Neural Networks and Machine Learning," *Springer Science and Business Media*, 2017.

[3] G. Rysbayeva, "Sequence Recommendation based on Deep Learning," *International Journal of Advanced Computer Science and Applications*, 2023.

[4] "Advances in Knowledge Discovery and Data Mining," *Springer Science and Business Media LLC*, 2020.

[5] K. A. L. Hernandez, "Deep Learning in Spatiotemporal Cardiac Imaging: A Review of Methodologies and Clinical Usability," 2020.

[6] N. Pratap, "Retinal blood vessels segmentation by using Gumbel probability distribution function based matched filter," 2016.

[7] E. A, "Deep learning model for fully automated breast cancer detection system from thermograms," 2022.

[8] K. J. Joseph, "C4Synth: Cross-Caption Cycle-Consistent Text-to-Image Synthesis," *2019 IEEE Winter Conference on Applications of Computer Vision (WACV),* 2019.

[9] "Knowledge Science, Engineering and Management," *Springer Science and Business Media LLC,* 2021.

[10] V. Morkun, "Application of Magnetic and Ultrasonic Methods for Determining Parameters of Ferromagnetic Components in Iron Ore Slurry Flows," 2021.

[11] S. Krig., "Computer Vision Metrics",," *Springer Science and Business Media LLC,* 2016.

[12] T. Lee, "Deep Learning for Hydrometeorology and Environmental Science," *Springer Science and Business Media LLC,* 2021.

[13] "ICT Analysis and Applications," *Springer Science and Business Media LLC,* 2021.

[14] "Computer Vision – ECCV 2020," *Springer Science and Business Media LLC,* , 2020.