A STUDY ON CONDITIONAL GENERATIVE ADVERSARIAL NETWORK USING RETINAL FUNDUS IMAGES

A Project
Submitted in partial fulfillment of the requirements for
the award of the Degree of

MASTER OF COMPUTER APPLICATION
By

ASHUTOSH CHOUDHURY
ROLL NO.: 12022010010018 AND REGISTRATION NO.: 221040510010

LOKNATH GHOSH
ROLL NO.: 12022010010055 AND REGISTRATION NO.: 221040510020

TUSHAR SARKAR
ROLL NO.: 12022010010027 AND REGISTRATION NO.: 221040510057

RAHUL MAJUMDER
ROLL NO.: 12022010010011 AND REGISTRATION NO.: 221040510030

SAYAN MUKHERJEE
ROLL NO.: 12022010010010 AND REGISTRATION NO.: 221040510046

MAULANA ABUL KALAM AZAD
UNIVERSITY OF TECHNOLOGY,
WEST BENGAL

DEPARTMENT OF COMPUTER APPLICATION

INSTITUTE OF ENGINEERING AND MANAGEMENT, 2024

DECLARATION CERTIFICATE

This is to certify that the work presented in the thesis entitled **“A Study on Conditional Generative Adversarial Network using Retinal Fundus Images”** in partial fulfillment of the requirement for the award of degree of **Master of Computer Application** of Institute of Engineering & Management is an authentic work carried out under my supervision and guidance.

To the best of my knowledge the content of this thesis does not form a basis for the award of any previous Degree to anyone else.

DATE : 10-04-2024

(Supratim Ghosh)

Dept. of Computer Application
Institute of Engineering & Management

Head of the Department
Dept. of Computer Application and Science
Institute of Engineering & Management

CERTIFICATE OF APPROVAL

The foregoing thesis entitled **"A Study on Conditional Generative Adversarial Network using Retinal Fundus Images"** is hereby approved as a creditable study of research topic and has been presented in satisfactory manner to warrant its acceptance as prerequisite to the degree for which it has been submitted.

It is understood that by this approval, the undersigned do not necessarily endorse any conclusion drawn or opinion expressed therein, but approve the thesis for the purpose for which it is submitted.

Viva-Voice held on ___________

(Internal Examiner)	**(External Examiner)**

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our Guide, **Supratim Ghosh** who helped us a lot in this project, his valuable suggestions helped us to solve tough challenges and without his help this project could not have been completed in time.

A special thanks to our Head of Department who gave us the golden opportunity to do this wonderful project on the topic **“A Study on Conditional Generative Adversarial Network using Retinal Fundus Images”**, which helped us to gain a significant knowledge in the aforesaid subjects. Secondly, we would like to thank our friends who helped us a lot in finalizing this project within the given time frame.

Name of student: Ashutosh Choudhury
Roll No.: 12022010010018

Name of student: Loknath Ghosh
Roll No.: 12022010010055

Name of student: Tushar Sarkar
Roll No.: 12022010010027

Name of student: Rahul Majumder
Roll No.: 12022010010011

Name of student: Sayan Mukherjee
Roll No.: 12022010010010

DECLARATION

We hereby declare that this project report is titled **“A Study on Conditional Generative Adversarial Network using Retinal Fundus Images”** is a genuine project work carried out by us, in **MCA(Master Of Computer and Application)** degree course of **INSTITUITE OF ENGINEERING AND MANAGEMENT, Kolkata** and has not been submitted to any other course or university for the award of our degree by us.

SIGNATURE

PREFACE

In the realm of healthcare, technological advancements have always been pivotal in enhancing diagnostic capabilities and improving patient outcomes. Among these advancements, the integration of artificial intelligence (AI) and machine learning (ML) has emerged as a transformative force, offering unprecedented opportunities to revolutionize medical imaging and diagnosis.

In the domain of ophthalmology, retinal fundus photography serves as a cornerstone in the early detection and management of various ocular pathologies, ranging from diabetic retinopathy to glaucoma. However, the interpretation of retinal fundus images often relies heavily on the expertise of trained ophthalmologists, leading to challenges related to scalability, accessibility, and efficiency in healthcare delivery.

To address these challenges, the development of generative models for retinal funduscopy represents a significant advancement in augmenting clinical workflows and expanding the reach of eye care services. By harnessing the power of deep learning and generative adversarial networks (GANs), these models demonstrate remarkable capabilities in synthesizing realistic retinal fundus images, offering invaluable support to healthcare professionals in diagnosis, education, and research.

This preface serves as an introduction to the innovative work presented within this volume, exploring the evolution, applications, and implications of generative models for retinal funduscopy. Through a collaboration of multidisciplinary expertise spanning computer science, medicine, and engineering, this collection endeavors to shed light on the transformative potential of AI-driven approaches in ophthalmic imaging.

As we embark on this journey of exploration and discovery, let us remain mindful of the ethical considerations, regulatory frameworks, and patient-centric principles that underpin our pursuit of technological innovation in healthcare. May the insights shared within these pages inspire further dialogue, collaboration, and innovation towards a future where advanced AI-driven solutions empower healthcare providers and enhance patient care.

In addition, a comprehensive study on Generative Adversarial Networks (GANs) utilizing Retinal Fundus images as a focal point promises to yield profound insights into the capabilities and limitations of these AI-driven approaches. Such a study would explore various aspects including data acquisition, preprocessing, model architecture design, training methodologies, and performance evaluation metrics specific to retinal imaging tasks. By elucidating the intricacies of GANs in this context, researchers can uncover new avenues for advancing retinal imaging technology, ultimately leading to improved diagnostic accuracy, personalized treatment strategies, and better patient outcomes in the field of ophthalmology.

CONTENTS

Chapter 1: INTRODUCTION

In recent years, the convergence of artificial intelligence (AI) and medical imaging has sparked remarkable progress in healthcare. Among these breakthroughs, the emergence of generative models for retinal funduscopy represents a notable frontier in ophthalmology, offering profound implications for the diagnosis and management of ocular diseases.

Retinal fundus photography, a non-invasive imaging modality, plays a critical role in early disease detection and monitoring across various retinal pathologies such as diabetic retinopathy, age-related macular degeneration, and glaucoma. However, the interpretation of fundus images traditionally relies on the expertise of skilled ophthalmologists, posing challenges related to scalability, accessibility, and diagnostic consistency, particularly in underserved areas.

Generative models, particularly generative adversarial networks (GANs), have emerged as a potent tool for synthesizing realistic images that closely approximate real-world data distributions. In the realm of retinal funduscopy, these models hold the potential to generate high-fidelity synthetic fundus images, mirroring those captured through clinical imaging systems.

The adoption of generative models in retinal funduscopy offers several compelling advantages. Firstly, it facilitates the creation of expansive, diverse datasets, addressing the scarcity of annotated images and enabling robust training of deep learning algorithms. Secondly, it enables dataset augmentation with variations in pathology, demographics, and imaging conditions, enhancing the generalization capabilities of AI models. Moreover, generative models can serve as valuable aids for data augmentation, regularization, and domain adaptation, thereby enhancing the performance and resilience of diagnostic algorithms.

Beyond diagnostic assistance, generative models for retinal funduscopy hold promise in educational settings, where they can generate simulated cases for training and evaluation. Additionally, they provide a platform for exploring hypothetical scenarios, studying disease progression, and assessing treatment impacts in controlled environments.

However, the integration of generative models into clinical practice necessitates addressing several challenges and ethical considerations. These include ensuring transparency, interpretability, and generalizability of AI models, as well as addressing concerns related to data privacy, security, and regulatory compliance.

In this volume, we delve into the evolution, methodologies, applications, and implications of generative models for retinal funduscopy. Through interdisciplinary collaboration bridging computer science, medicine, and engineering, we aim to provide insights, perspectives, and practical guidance for researchers, clinicians, and stakeholders navigating this rapidly evolving landscape.

Together, we explore the transformative potential of generative models in reshaping the future of retinal imaging and advancing the frontiers of ophthalmic care. Additionally, we will conduct a comprehensive study on Generative Adversarial Networks (GANs) utilizing Retinal Fundus images, focusing on aspects such as data acquisition, preprocessing, model architecture, training methodologies, and performance evaluation metrics specific to retinal imaging tasks. This study will shed light on the capabilities and limitations of GANs in retinal imaging, paving the way for enhanced diagnostic accuracy, personalized treatment strategies, and improved patient outcomes in ophthalmology.

Chapter 2: BACKGROUND STUDY

2.1 Bernstein Polynomial

In the mathematical field of numerical analysis, a Bernstein polynomial is a polynomial expressed as a linear combination of Bernstein basis polynomials. The idea is named after mathematician Sergei Natanovich Bernstein.

Bernstein basis polynomials:

The n+1 Bernstein basis polynomials of degree **n** are defined as:

$$b_{v,n}(x) = \binom{n}{v} x^v (1-x)^{n-v}, v = 0, \ldots, n, \qquad \ldots(1)$$

where $\binom{n}{v}$ is a binomial coefficient.

So, for example

$$b_{2,5}(x) = \binom{5}{2} x^2 (1-x)^3 = 10x^2 (1-x)^3 \qquad \ldots(2)$$

The first few Bernstein basis polynomials for blending 1, 2, 3 or 4 values together are:

$$\begin{array}{llll}
b_{0,0}(x) = 1, & & & \\
b_{0,1}(x) = 1 - x, & b_{1,1}(x) = x & & \\
b_{0,2}(x) = (1-x)^2, & b_{1,2}(x) = 2x(1-x), & b_{2,2}(x) = x^2 & \\
b_{0,3}(x) = (1-x)^3, & b_{1,3}(x) = 3x(1-x)^2, & b_{2,3}(x) = 3x^2(1-x), & b_{3,3}(x) = x^3
\end{array}$$

The Bernstein basis polynomials of degree **n** form a basis for the vector space of polynomials of degree at most **n** with real coefficients.

One potential application of Bernstein polynomials in retinal fundoscopy is in the representation and analysis of retinal vasculature. The blood vessels in the retina exhibit intricate patterns and structures that are crucial indicators of various ocular diseases such as diabetic retinopathy and hypertensive retinopathy. By employing Bernstein polynomials, it may be possible to model the shapes and trajectories of retinal blood vessels with high accuracy, enabling quantitative analysis of vessel tortuosity, branching patterns, and other morphological features.

Furthermore, Bernstein polynomials could be utilized in image enhancement and denoising algorithms for retinal fundus images. By representing the image data as a polynomial function, it becomes possible to apply mathematical operations such as differentiation, integration, and smoothing to enhance image clarity, reduce noise, and improve overall image quality. This can be particularly beneficial in improving the visibility of subtle pathological features and enhancing diagnostic accuracy.

Overall, the integration of Bernstein polynomials into retinal fundoscopy holds promise for advancing our understanding of retinal anatomy, pathology, and function. By leveraging the mathematical properties and computational capabilities of Bernstein polynomials, researchers and clinicians can develop innovative approaches for image analysis, diagnosis, and treatment monitoring in ophthalmology.

2.2 K-Means Clustering

K-means clustering is a popular unsupervised machine learning algorithm used for partitioning a dataset into a predefined number of clusters. The algorithm iteratively assigns data points to the nearest cluster centroid and then updates the centroids based on the mean of the data points assigned to each cluster. This process continues until the centroids no longer change significantly or a maximum number of iterations is reached.

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

(a) Determines the best value for K center points or centroids by an iterative process.
(b) Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has data points with some commonalities, and it is away from other clusters.

The working of K-means Clustering Algorithm is shown in Figure 2.1.

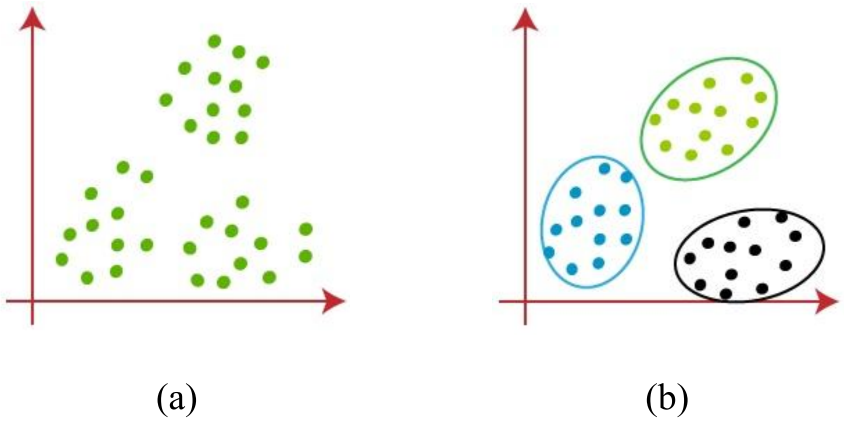


Figure 2.1 *: (a) Graph showing a group of points before K-means.*
(b) Graph showing cluster of points after K-means.

2.3 GENERATIVE ADVERSARIAL NETWORKS

GANs are a framework for teaching a deep learning model to capture the training data distribution so we can generate new data from that same distribution. GANs were invented by Ian Goodfellow in 2014 and first described in the paper Generative Adversarial Nets. They are made of two distinct models, a generator and a discriminator. The job of the generator is to spawn 'fake' images that look like the training images. The job of the discriminator is to look at an image and output whether or not it is a real training image or a fake image from the generator. During training, the generator is constantly trying to outsmart the discriminator by generating better and better fakes, while the discriminator is working to become a better detective and correctly classify the real and fake images. The equilibrium of this game is when the generator is generating perfect fakes that look as if they came directly from the training data, and the discriminator is left to always guess at 50% confidence that the generator output is real or fake.

Now, lets define some notation to be used throughout tutorial starting with the discriminator. Let x be data representing an image. D(x) is the discriminator network which outputs the (scalar) probability that x came from training data rather than the generator. Here, since we are dealing with images, the input to D(x) is an image of CHW size 3x64x64. Intuitively, D(x) should be HIGH when x comes from training data and LOW when x comes from the generator. D(x) can also be thought of as a traditional binary classifier.

For the generator's notation, let z be a latent space vector sampled from a standard normal distribution. G(z) represents the generator function which maps the latent vector z to data-space. The goal of G is to estimate the distribution that the training data comes from (p_{data}) so it can generate fake samples from that estimated distribution (p_g).

So, D(G(z)) is the probability (scalar) that the output of the generator G is a real image. As described in Goodfellow's paper, D and G play a minimax game in which D tries to maximize the probability it correctly classifies reals and fakes (logD(x)), and G tries to minimize the probability that D will predict its outputs are fake (log(1−D(G(z)))). From the paper, the GAN loss function is

$$\min_{G} \max_{D} V\,(D, G) = E_{x\sim p_{data}(x)}\,[logD(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))]$$

…(3)

In theory, the solution to this minimax game is where p_g=p_{data} , and the discriminator guesses randomly if the inputs are real or fake. However, the convergence theory of GANs is still being actively researched and in reality models do not always train to this point.

2.4 DCGAN

DCGAN or Deep Convolutional Generative Adversarial Network, is a type of generative model that uses convolutional neural networks (CNNs) for both the generator and discriminator networks in a GAN framework. DCGANs were introduced by Radford et al. in 2015 and have since become a popular architecture for generating high-quality synthetic images.

The key features and components of DCGANs include:

1. Generator Network: The generator takes random noise as input and generates synthetic images. It typically consists of several convolutional layers followed by upsampling layers (such as transposed convolutions or upsampling followed by convolutions) to progressively transform the input noise into a realistic image. Batch normalization and ReLU activation functions are commonly used in the generator to stabilize training and encourage feature diversity.

2. Discriminator Network: The discriminator network distinguishes between real and fake images. It takes images as input and outputs a probability indicating the likelihood that the input image is real. Similar to the generator, the discriminator consists of convolutional layers followed by downsampling layers (such as max pooling) to extract features from the input images. Batch normalization and LeakyReLU activation functions are often used in the discriminator to improve training stability and prevent vanishing gradients.

3. Adversarial Training: DCGANs are trained using an adversarial training scheme, where the generator and discriminator networks are trained simultaneously in a min-max game. The generator aims to generate images that are indistinguishable from real images, while the discriminator aims to accurately classify real and fake images. The training process involves iteratively updating the parameters of both networks to improve their performance.

4. Loss Functions: The generator and discriminator networks are trained using different loss functions. The generator is trained to minimize the binary cross-entropy loss between the discriminator's predictions and a label indicating that the generated images are real. Conversely, the discriminator is trained to minimize the binary cross-entropy loss between its predictions and the true labels (real or fake) of the input images.

5. Training Stability Techniques: Several techniques are used to improve the stability and convergence of DCGAN training, including weight initialization strategies, batch normalization, leaky ReLU activation functions, and gradient clipping.

DCGANs have demonstrated impressive results in generating realistic images across various domains, including natural images, faces, and artworks. They have been used in applications such as image synthesis, data augmentation, image-to-image translation, and style transfer. However, training DCGANs can be challenging and requires careful tuning of hyperparameters and network architectures to achieve good results.

A DCGAN is a direct extension of the GAN described above, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively. It was first described by Radford et. al. in the paper Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks. The discriminator is made up of strided convolution layers, batch norm layers, and LeakyReLU activations. The input is a 3x64x64 input image and the output is a scalar probability that the input is from the real data distribution. The generator is comprised of convolutional-transpose layers, batch norm layers, and ReLU activations. The input is a latent vector, z, that is drawn from a standard normal distribution and the output is a 3x64x64 RGB image. The strided conv-transpose layers allow the latent vector to be transformed into a volume with the same shape as an image. In the paper, the authors also give some tips about how to setup the optimizers, how to calculate the loss functions, and how to initialize the model weights, all of which will be explained in the coming sections.

GENERATOR

The generator, G, is designed to map the latent space vector (z) to data-space. Since our data are images, converting z to data-space means ultimately creating a RGB image with the same size as the training images (i.e. 3x64x64). In practice, this is accomplished through a series of strided two dimensional convolutional transpose layers, each paired with a 2d batch norm layer and a relu activation. The output of the generator is fed through a tanh function to return it to the input data range of [−1,1]. It is worth noting the existence of the batch norm functions after the conv-transpose layers, as this is a critical contribution of the DCGAN paper. These layers help with the flow of gradients during training. An image of the generator is shown below in Figure 2.2.

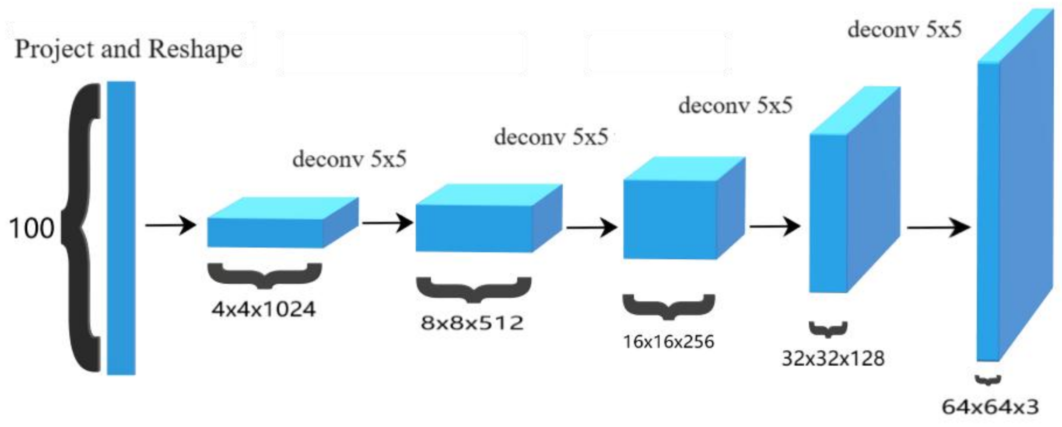


Figure 2.2 *: Generator*

DISCRIMINATOR

As mentioned, the discriminator, D, is a binary classification network that takes an image as input and outputs a scalar probability that the input image is real (as opposed to fake). Here, D takes a 3x64x64 input image, processes it through a series of Conv2d, BatchNorm2d, and LeakyReLU layers, and outputs the final probability through a Sigmoid activation function. This architecture can be extended with more layers if necessary for the problem, but there is significance to the use of the strided convolution, BatchNorm, and LeakyReLUs. The DCGAN paper mentions it is a good practice to use strided convolution rather than pooling to downsample because it lets the network learn its own pooling function. Also batch norm and leaky relu functions promote healthy gradient flow which is critical for the learning process of both G and D. The Figure 2.3 below shows the architecture of the discriminator.

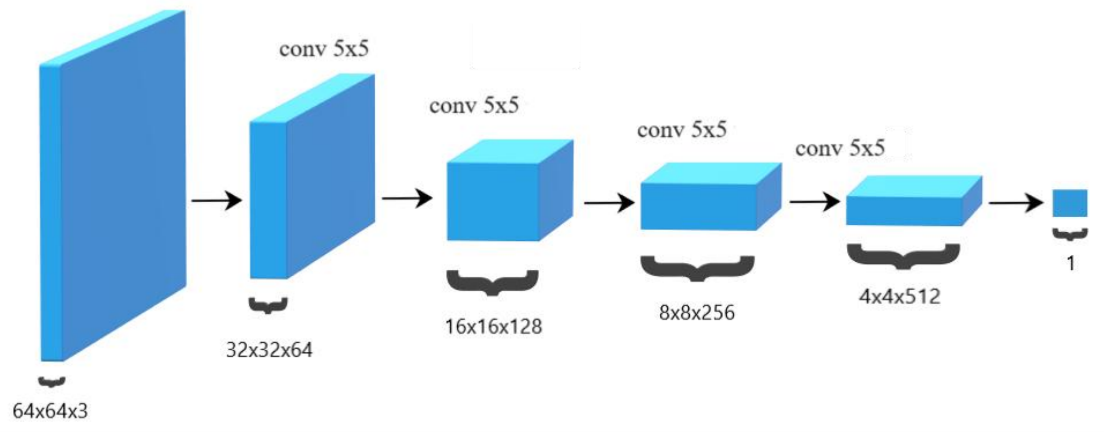


Figure 2.3 *: Discriminator*

2.5 Activation Functions

In artificial neural networks, an activation function is one that outputs a smaller value for tiny inputs and a higher value if its inputs are greater than a threshold. An activation function "fires" if the inputs are big enough; otherwise, nothing happens. An activation function, then, is a gate that verifies how an incoming value is higher than a threshold value.

Because they introduce non-linearities in neural networks and enable the neural networks can learn powerful operations, activation functions are helpful. A feedforward neural network might be refactored into a straightforward linear function or matrix transformation on to its input if indeed the activation functions were taken out.

By generating a weighted total and then including bias with it, the activation function determines whether a neuron should be turned on. The activation function seeks to boost a neuron's output's nonlinearity.

Explanation: As we are aware, neurons in neural networks operate in accordance with weight, bias, and their corresponding activation functions. Based on the mistake, the values of the neurons inside a neural network would be modified. This process is known as back-propagation. Back-propagation is made possible by activation functions since they provide the gradients and error required to change the biases and weights.

Need of Non-linear Activation Functions

An interconnected regression model without an activation function is all that a neural network is. Input is transformed nonlinearly by the activation function, allowing the system to learn and perform more challenging tasks.

It is merely a thing procedure that is used to obtain a node's output. It also goes by the name Transfer Function.

The mixture of two linear functions yields a linear function, so no matter how several hidden layers we add to a neural network, they all will behave in the same way. The neuron cannot learn if all it has is a linear model. It will be able to learn based on the difference with respect to error with a non-linear activation function.

The mixture of two linear functions yields a linear function in itself, so no matter how several hidden layers we add to a neural network, they all will behave in the same way. The neuron cannot learn if all it has is a linear model.

The two main categories of activation functions are:

1. Linear Activation Function
2. Non-linear Activation Functions

Linear Activation Function

As can be observed from the figure 2.4, the functional is linear or linear. Therefore, no region will be employed to restrict the functions' output.

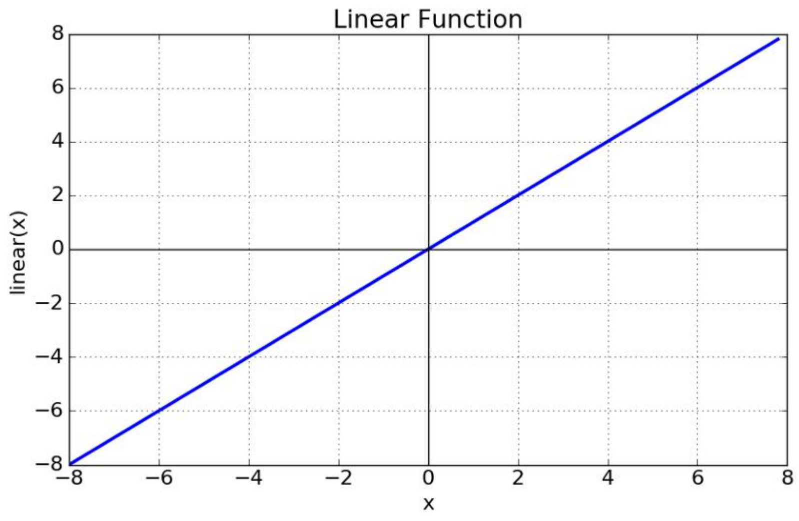


Figure 2.4: Linear Activation Function

It doesn't help with the complexity or various parameters of usual data that is fed to the neural networks.

Non-linear Activation Function

The Nonlinear Activation Functions are the most used activation functions. Non-linearity helps to makes the graph look something like the figure 2.5 below

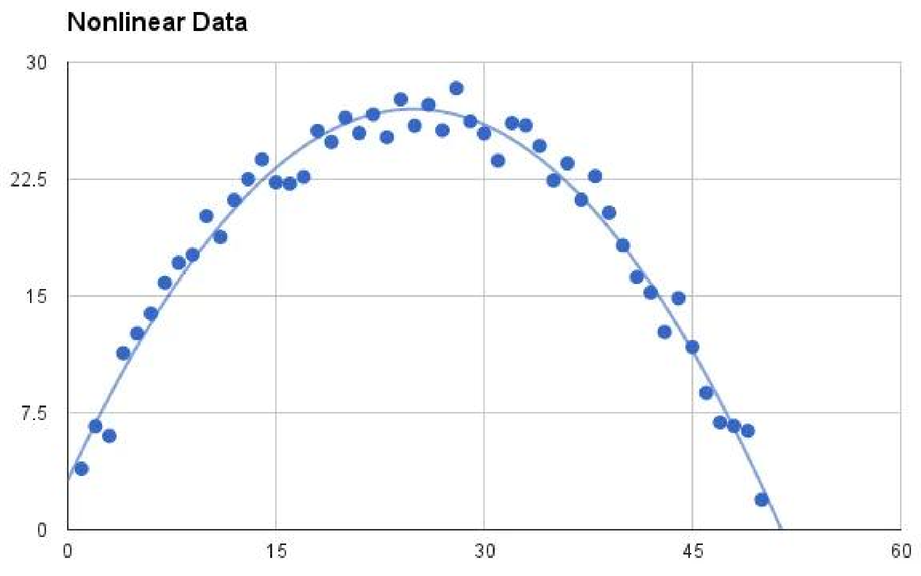


Figure 2.5: Non-linear Activation Function

It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.

The main terminologies needed to understand for nonlinear functions are:

Derivative or Differential: Change in y-axis w.r.t. change in x-axis.It is also known as slope.

Monotonic function: A function which is either entirely non-increasing or non-decreasing.

The Nonlinear Activation Functions are mainly divided on the basis of their range or curves-

1. Sigmoid or Logistic Activation Function

The Sigmoid Function curve looks like a S-shape as shown in figure 2.6.

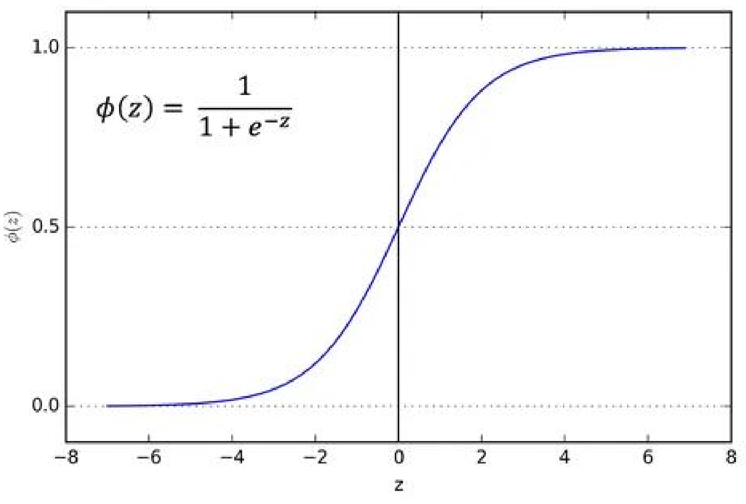


Figure 2.6: Sigmoid Function

The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output.Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

The function is differentiable.That means, we can find the slope of the sigmoid curve at any two points.

The function is monotonic but function’s derivative is not.

The logistic sigmoid function can cause a neural network to get stuck at the training time.

2. Tanh or hyperbolic tangent Activation Function

tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped) as shown in figure 2.7.

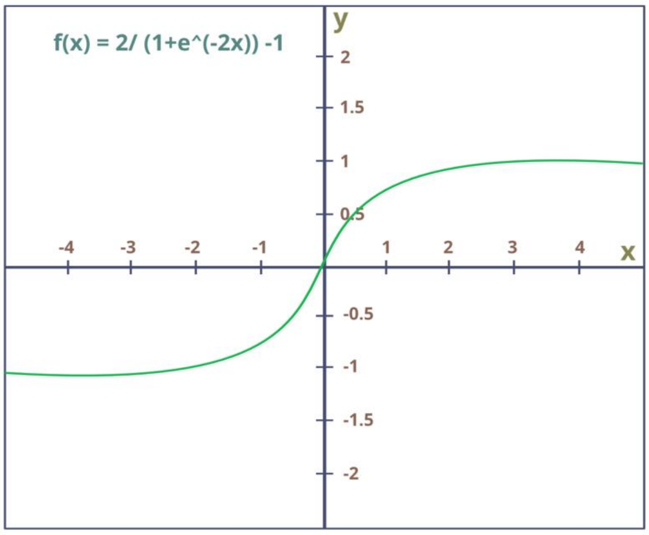


Figure 2.7: tanh Function

The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

The function is differentiable.

The function is monotonic while its derivative is not monotonic.

The tanh function is mainly used classification between two classes.

$$\mathrm{f(x) = tanh(x)} = \frac{2}{1+e^{-2x}} - 1 \qquad \ldots(4)$$

3. ReLU (Rectified Linear Unit) Activation Function

The ReLU is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning.

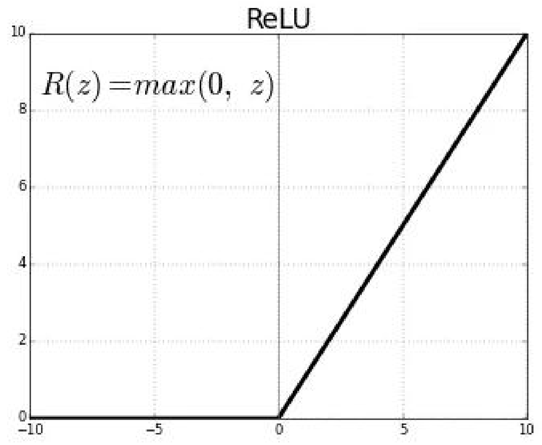


Figure 2.8: ReLu Function

As we can see in figure 2.8, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

Range: [0 to infinity)

The function and its derivative both are monotonic.

But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

4. Leaky ReLU

It is the improved version of the ReLU. It converts negative values to make them close to 0 but not actually 0, solving the dying ReLU issue that arises from using the standard ReLU function.

The "dying ReLU" problem refers to a situation when the input to a ReLU neuron is consistently negative, causing the neuron to always output zero. During the backpropagation phase of training, if the gradients flowing through that neuron are consistently zero, the weights connected to that neuron may not be updated, and the neuron essentially becomes "dead" or "dormant." Once a neuron is in this state, it will continue to output zero for all inputs, and it won't contribute to the learning process.

It is an attempt to solve the dying ReLU problem. The figure 2.9 shows the comparison of ReLu and Leaky ReLu for better understanding.

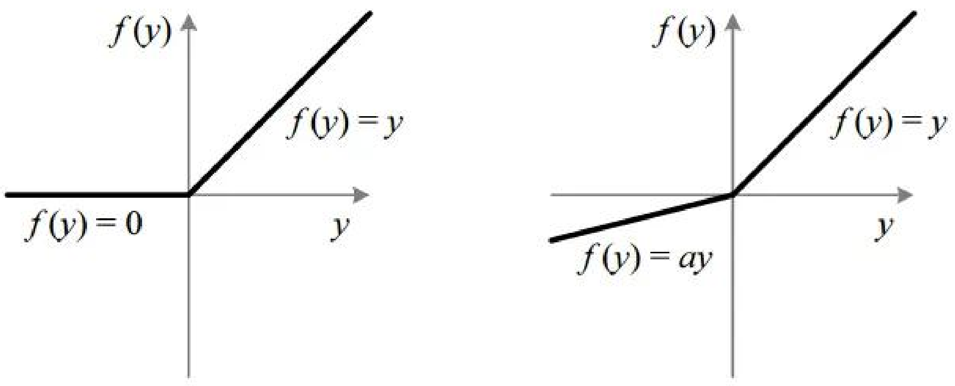


Figure 2.9: ReLu v/s Leaky ReLu

The leak helps to increase the range of the ReLU function. Usually, the value of a is 0.01 or so.

When a is not 0.01 then it is called Randomized ReLU.

Therefore the range of the Leaky ReLU is (-infinity to infinity).

Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

2.6 ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks

ESRGAN, or Enhanced Super-Resolution Generative Adversarial Networks, is a cutting-edge model designed to reconstruct high-resolution (HR) images or sequences from lower-resolution (LR) observations. This technology is particularly useful in upscaling images, for example, transforming a 720p image into a 1080p one. ESRGAN employs deep convolutional neural networks to recover HR images from LR ones, with the generator network learning to create realistic images and the discriminator network learning to differentiate between real and generated images. Through a process of competition and feedback, the generator network improves its ability to create high-quality images.

The technical architecture of ESRGAN is based on SRResNet, with residual-in-residual blocks. It uses a mixture of context, perceptual, and adversarial losses. Context and perceptual losses are used for proper image upscaling, while the adversarial loss pushes the neural network towards the natural image manifold. This is achieved using a discriminator network that is trained to differentiate between the super-resolved images and original photo-realistic images.

One of the key advantages of ESRGAN is its ability to generate high-quality, realistic images from lower resolution inputs. This is achieved through the use of a generative adversarial network, which continually improves the quality of the generated images through a process of competition and feedback. This makes ESRGAN an excellent tool for a variety of applications where image quality is paramount.

In few words, image super-resolution (SR) techniques reconstruct a higher-resolution (HR) image or sequence from the observed lower-resolution (LR) images, e.g. upscaling of 720p image into 1080p.

One of the common approaches to solving this task is to use deep convolutional neural networks capable of recovering HR images from LR ones. And ESRGAN (Enhanced SRGAN) is one of them.

Key points of ESRGAN:

1. SRResNet-based architecture with residual-in-residual blocks;
2. Mixture of context, perceptual, and adversarial losses. Context and perceptual losses are used for proper image upscaling, while adversarial loss pushes neural network to the natural image manifold using a discriminator network that is trained to differentiate between the super-resolved images and original photo-realistic images.

The architecture of ESRGAN is shown in the figure 2.10 below.

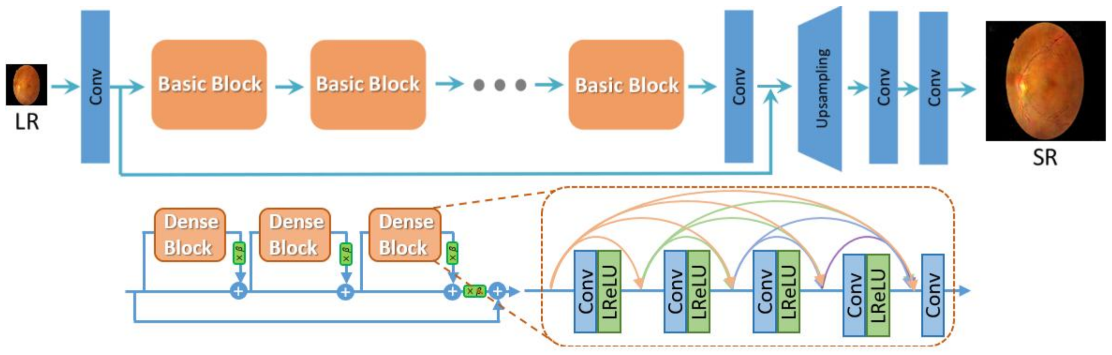


Figure 2.10: ESRGAN Architecture

Chapter 3: DATA SOURCE

We have Digital Retinal Images for Vessel Extraction Dataset with a sample training set image.

The database enables comparative studies on segmentation of blood vessels in retinal images. Automatic detection and analysis of the vasculature can assist in the implementation of screening programs for diabetic retinopathy. Automatic generation of retinal maps and extraction of branch points have been used for temporal or multimodal image registration and retinal image mosaic synthesis. Moreover, the retinal vascular tree is found to be unique for each individual and can be used for biometric identification.

The DRIVE database has been established to enable comparative studies on segmentation of blood vessels in retinal images. Retinal vessel segmentation and delineation of morphological attributes of retinal blood vessels, such as length, width, tortuosity, branching patterns and angles are utilized for the diagnosis, screening, treatment, and evaluation of various cardiovascular and ophthalmologic diseases such as diabetes, hypertension, arteriosclerosis and chorodial neovascularization. Automatic detection and analysis of the vasculature can assist in the implementation of screening programs for diabetic retinopathy, can aid research on the relationship between vessel tortuosity and hypertensive retinopathy, vessel diameter measurement in relation with diagnosis of hypertension, and computer-assisted laser surgery. Automatic generation of retinal maps and extraction of branch points have been used for temporal or multimodal image registration and retinal image mosaic synthesis. Moreover, the retinal vascular tree is found to be unique for each individual and can be used for biometric identification.

The database contains the photographs from a diabetic retinopathy screening program in Netherlands. The screening population consisted of 400 diabetic subjects between 25-90 years of age. Forty photographs have been randomly selected, 33 do not show any sign of diabetic retinopathy and 7 show signs of mild early diabetic retinopathy. Each image has been JPEG compressed.

The images were acquired using a Canon CR5 non-mydriatic 3CCD camera with a 45 degree field of view (FOV). Each image was captured using 8 bits per color plane at 768 by 584 pixels. The FOV of each image is circular with a diameter of approximately 540 pixels. For this database, the images have been cropped around the FOV. For each image, a mask image is provided that delineates the FOV.

The set of 40 images has been divided into a training and a test set, both containing 20 images. For the training images, a single manual segmentation of the vasculature is available. For the test cases, two manual segmentations are available; one is used as gold standard, the other one can be used to compare computer generated segmentations with those of an independent human observer. Furthermore, a mask image is available for every retinal image, indicating the region of interest. All human observers that manually segmented the vasculature were instructed and trained by an experienced ophthalmologist. They were asked to mark all pixels for which they were for at least 70% certain that they were vessel.

Chapter 4: PROPOSED METHODOLOGY

Our initial step is to remove the blood vessels such that a clear retinal image is obtained without the blood vessels. The obtained image where the vessels are removed or discarded needs to be filled with some colour, which is our main objective. To fill the vessel portion we use a specific mathematical expression called the Bernstein Polynomial.

We split the the image into three channels BGR and we take the green channel (here) and the required operations are performed on it and the resultant filled green channel image is formed.

The dataset of three images consisting of the ground truth, retinal image and mask shown in Figure 4.1 is used to first generate an image where the blood vessels are removed in each channel.

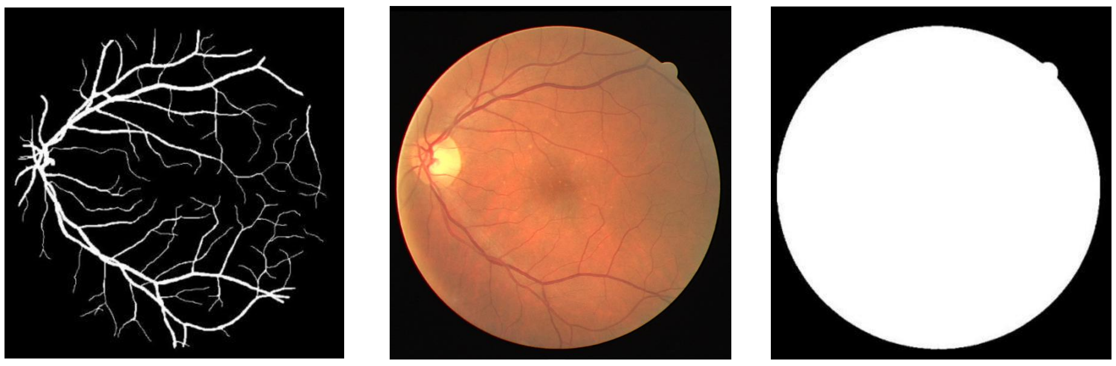

(a) (b) (c)

***Figure 4.1** : (a) Retinal image of the ground truth. (b) The original input image. (c) The mask image.*

The green channel image and the ground truth image are compared and the pixels locations that have a value of more than 127 in the ground truth image; the same locations in the green channel retinal image are changed to a value of 0 that is the color black. Hence, through this procedure we get an image where only the blood vessels are highlighted in black color which makes it easier for selection henceforward.

After the required operations are performed we get the resultant green channel image with discarded vessels as shown in Figure 4.2.

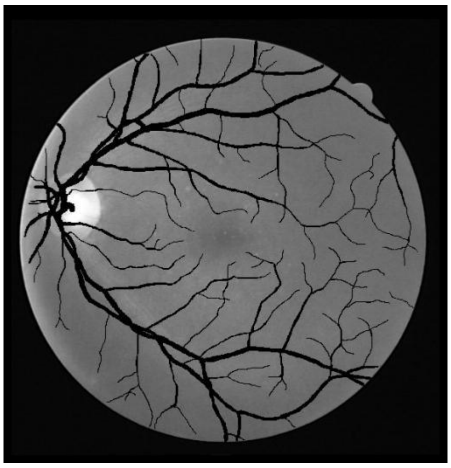


Figure 4.2 *: Green channel image with black coloured blood vessels.*

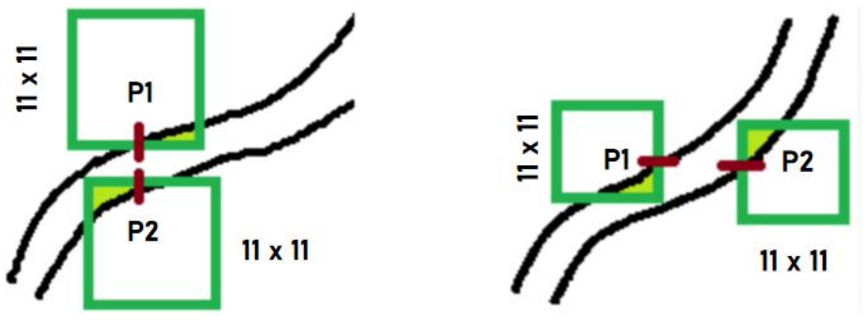
In the next step we compare the mask image with the new green channel image obtained to get only the retinal portion of the image excluding the black background. This makes it easier to search for the black colored blood vessels.

Now, we have got the starting pixel index and the ending pixel index of the blood vessels. We use a defined logic to get P1 and P2 values through which after applying them in the Bernstein polynomial equation we can fill up the blood vessel pixels.

Taking the neighbouring pixels of the black pixels we apply Bernstein equation and produce two images: one an image with horizontal traversal of pixel points and one with the vertical traversal of pixel points as shown in Figure 4.3. Merging both the images will produce an image where the blood vessels are filled.

The logic used is creating a matrix window. We are creating two 11 x 11 matrices for neighbouring horizontal and vertical pixels of black pixels and then we are computing the average of the matrices which gives us P1 and P2. We apply this P1 and P2 values in the Bernstein equation.

Creating a matrix window gives us a more accurate average value that helps to fill the pixels points that are not coloured.



(a) (b)

Figure 4.3 *: (a) Vertical traversal by creating 11x11 matrix window to calculate P1 and P2.*
(b) Horizontal traversal by creating 11x11 matrix window to calculate P1 and P2.

During matrix creation the image may contain some white and black pixels as noise and hence we are taking those pixels and then take the average of the remaining pixels using the equation:

$$\sum \frac{I\,(k-x)}{(k-x)}$$

...(5)

where k is the matrix window size i.e k = 11 x 11 = 121 (here) and x is the number of black and white pixels we are ignoring.

After the blood vessels are completely filled we would get the resultant green channel image as shown in Figure 4.4. The filled vessels closely resemble the color of the rest of internal retinal image. This new green image generated is used to further progress through the project.

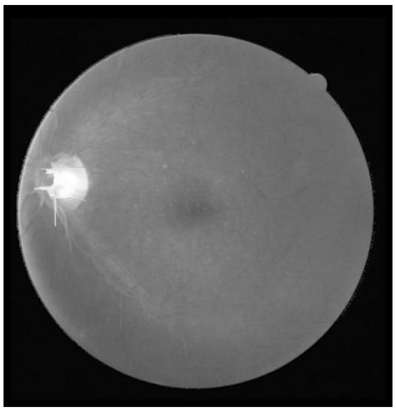
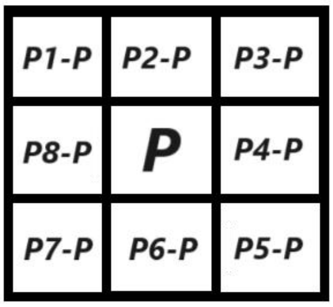


Figure 4.4 *: Green channel Image with filled blood vessels.*

Now, after getting the new green image we use Sklearn Python library to apply the K-means clustering algorithm. Through K-means clustering we perform region creation.

The aim is to create three clusters or groupings of retinal images which will highlight the optic disk region, high contrast region and low contrast region.

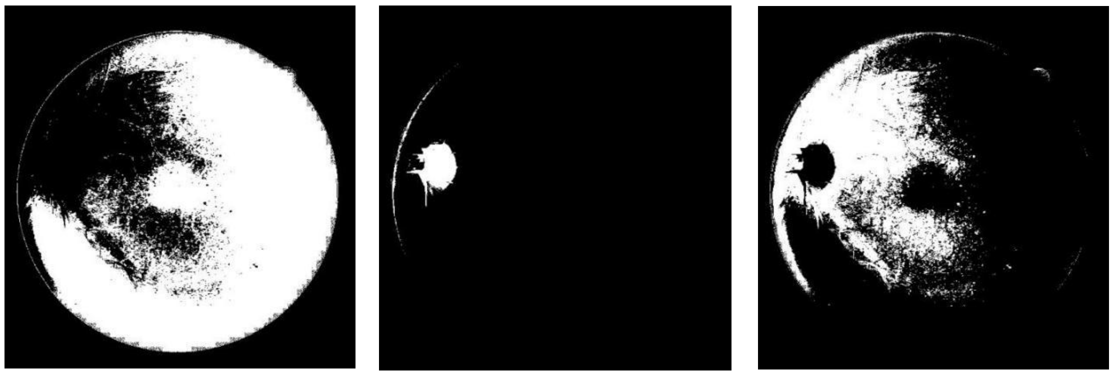
For feature extraction we take a 3x3 matrix window as shown in Figure 4.5 and by subtracting middle pixel value with the rest we take the minimum value i.e. <p, min> to add it as a feature.



P1-P	P2-P	P3-P
P8-P	P	P4-P
P7-P	P6-P	P5-P

Figure 4.5 *: Feature extraction using 3x3 matrix window.*

The features are used to create the three clusters (k=3). Now we compare the pixels of the new green channel image with each of the clusters to check which pixels points have the same value and then based upon that we create three new images which is our resultant optic disk region, high contrast region and low contrast region as shown in Figure 4.6.



(a) (b) (c)

Figure 4.6 *: (a) High Contrast Region. (b) Optic Disk Region. (c) Low Contrast Region.*

After the generation of the three regions we move on to the next phase of our project as we move on to train our GAN model. We generate only the optic region images from our retinal fundus dataset as here we are only concerned with optic disk region.

Once the optic disk region images are generated we pass the optic region dataset to the generator as henceforward it will generate RGB retinal images by learning and the retinal image dataset is passed through the discriminator as it will then be able to learn and identify the real and fake image generated by the generator.

We move on with our work and related fields of the project and they are defined as followed.

We put up a PyTorch environment for training a generative adversarial network (GAN) on retinal fundus images. This script imports necessary libraries, defines parameters, sets up data loading and pre-processing, and visualizes some training images as shown in Figure 4.1.

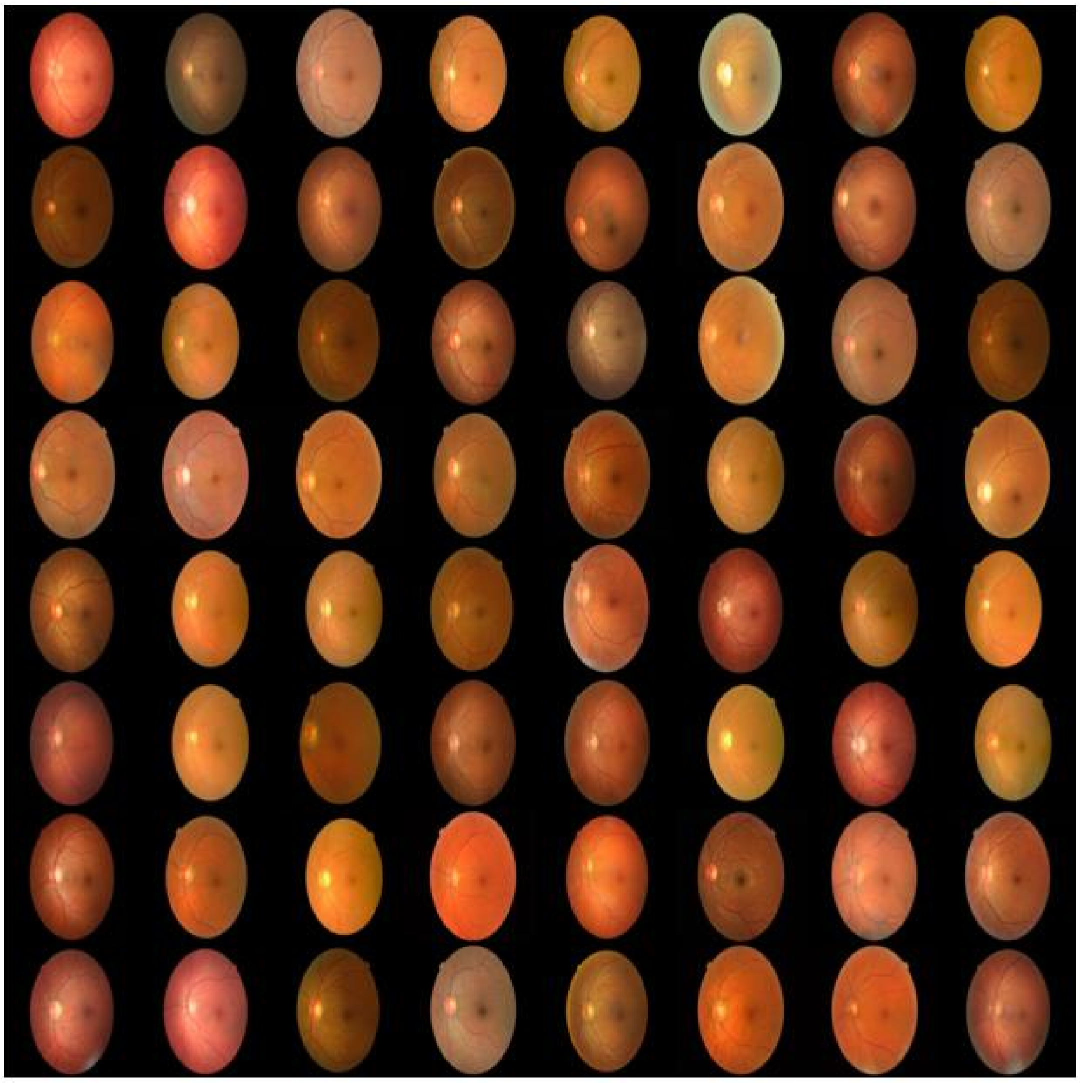


Figure 4.*7 : Training Images*

Thereon the optic disk batch input is clearly depicted in the figure 4.8.

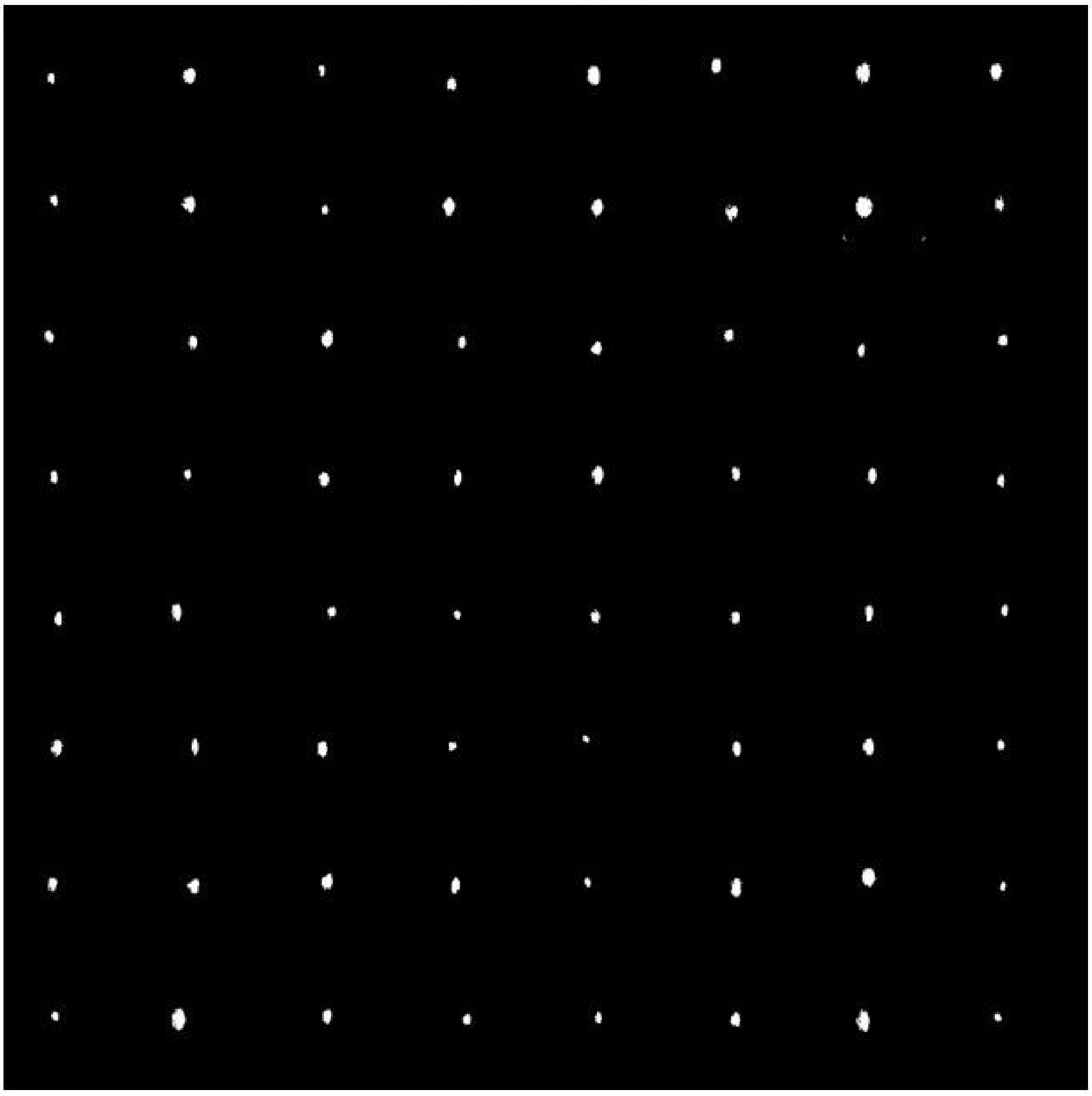


Figure 4.8 *: Training Optic Disk Images*

The project code outlines the architectural design for both the generator and discriminator networks within a Generative Adversarial Network (GAN) framework.

Firstly, the generator, encapsulated within the generator class, is tasked with synthesizing artificial images from random noise inputs. It comprises a sequence of transposed convolutional layers, effectively upscaling the input noise into a coherent image. Each convolutional layer is accompanied by batch normalization to stabilize training and rectified linear unit (ReLU) activation functions to introduce non-linearity. The final layer utilizes the hyperbolic tangent function (tanh) to ensure output pixel values fall within the range [-1, 1], which is typically conducive to image data normalization.

Conversely, the discriminator, embodied within the discriminator class, operates as an adversary to the generator. It scrutinizes both real and generated images, discerning between the two categories. The discriminator architecture comprises convolutional layers followed by batch normalization and leaky ReLU activation functions, a variation of traditional ReLU that allows a small, non-zero gradient when the input is negative. The final layer is governed by a sigmoid activation function, outputting a probability score indicating the likelihood of an input image being real.

In addition to defining the network architectures, the code includes weight initialization procedures for both networks and their forward pass implementations. Furthermore, it initializes the networks, netG for the generator and netD for the discriminator, and assigns them to the appropriate computational device, whether it be a GPU or CPU.

While the architectural setup is crucial, to train the GAN effectively, one would need to implement the training loop, encompassing processes such as optimization, loss calculation, and gradient updates. Moreover, fine-tuning hyperparameters and exploring alternative network architectures might be necessary to attain optimal performance and generate high-fidelity synthetic images.

The function display_images serves the purpose of visualizing a list of generated images. Upon invocation, it expects a list (img_list) containing batches of generated images, where each batch is represented as a PyTorch tensor. Inside the function, it extracts the last batch of images from the provided list using img_list[-1].

Subsequently, the function creates a new figure with a size of 8x8 inches, effectively setting up the canvas for visualization. It then turns off the axis labels to declutter the plot, enhancing the clarity of the displayed images. A title, "Fake Images," is set for the plot, providing context for the visualized content.

The core functionality of arranging the images into a grid is handled by vutils.make_grid(fake_batch.to(device), nrow=10, padding=2, normalize=True). This operation arranges the images into a grid, with the specified number of images displayed per row (nrow=10), along with padding between the images (padding=2). Additionally, it ensures normalization of pixel values (normalize=True), maintaining consistency in visual appearance.

Before displaying the grid of images, a final transformation is applied using np.transpose(..., (1, 2, 0)). This operation transposes the grid of images to conform to the expected format for visualization, which is Height x Width x Channels (HWC).

Finally, the plot containing the grid of fake images is displayed using plt.imshow(...) and plt.show(), providing a visual representation of the generated images. Overall, the function encapsulates the process of visualizing generated images in a clean and concise manner, facilitating inspection and analysis during the training process of generative models.

In the training phase the code initiates the training loop for the Generative Adversarial Network (GAN), encompassing the training of both the generator (netG) and discriminator (netD) networks. Here's a breakdown of the training process:

1. Loss Function and Optimizers Setup:

The Binary Cross Entropy (BCE) loss function is initialized (criterion = nn.BCELoss()), which computes the binary classification loss between predicted outputs and target labels.

Adam optimizers are configured for both the generator and discriminator networks (optimizerG and optimizerD, respectively), with specified learning rates and momentum parameters.

2. Training Loop:

The loop iterates over each epoch (for epoch in range(EPOCH_NUM)) and each batch of data (for i, data in enumerate(dataloader, 0)).

For each batch:

The discriminator is updated with real data by computing the loss on real images and backpropagating the gradients (errD_real.backward()).

Fake images are generated by passing random noise through the generator (noise = torch.randn(b_size, Z_DIM, 1, 1, device=device) and fake = netG(noise)).

The discriminator is updated with fake data by computing the loss on fake images and backpropagating the gradients (errD_fake.backward()).

The generator is updated based on the discriminator's response to the fake images (errG.backward()).

Training statistics such as discriminator loss (Loss_D), generator loss (Loss_G), discriminator's output on real images (D(x)), and discriminator's output on fake images (D(G(z))) are printed periodically.

Losses for both generator and discriminator are stored for visualization later (G_losses and D_losses).

3. Training Termination:

The training loop also includes conditions to monitor the average loss over the last few epochs. If certain thresholds are met (avg_G_Loss >= 20 or avg_D_Loss >= 5), or if the loss becomes zero, the training is terminated to prevent divergence.

4. Model Saving and Visualization:

Periodically, the trained generator and discriminator models are saved (torch.save(...)) to the specified directory.

The generated images are visualized using the display_images function, which arranges and displays a grid of fake images.

Overall, this orchestrates the training process of the GAN, ensuring the progressive improvement of both the generator and discriminator networks while guarding against instability or divergence during training. Additionally, it facilitates the monitoring and visualization of the training progress through loss values and generated images.

A loss graph is plotted that visualizes the training progress of both the generator (G) and discriminator (D) networks during training iterations. Here's an interpretation of the graph:

Generator Loss (G): The loss of the generator network measures how well it is able to deceive the discriminator by generating realistic images. As training progresses, the generator loss ideally decreases, indicating that the generator is improving in generating images that resemble real ones. A decreasing generator loss signifies that the generator is learning to produce more convincing images.

Discriminator Loss (D): The loss of the discriminator network measures its ability to distinguish between real and fake images. As training progresses, the discriminator loss ideally decreases, indicating that the discriminator is becoming better at distinguishing between real and fake images. A decreasing discriminator loss signifies that the discriminator is learning to differentiate between real and fake images more accurately.

The convergence and behavior of the loss curves provide insights into the training dynamics of the GAN:

Convergence: Ideally, the generator and discriminator losses both converge to stable values. Convergence indicates that the GAN has achieved a balance where the generator produces realistic images that can consistently fool the discriminator, and the discriminator becomes skilled at distinguishing between real and fake images.

Mode Collapse: If the generator loss decreases significantly while the discriminator loss remains high, it could indicate mode collapse, where the generator produces limited varieties of images that fool the discriminator. This situation typically arises when the generator exploits weaknesses in the discriminator rather than learning to produce diverse and realistic images.

Instability: Fluctuations or instability in the loss curves may occur during training, especially in the early stages. This instability can be addressed through techniques such as adjusting learning rates, using different optimization algorithms, or applying regularization techniques.

The generator and discriminator loss graph during training is depicted in figure 4.9.

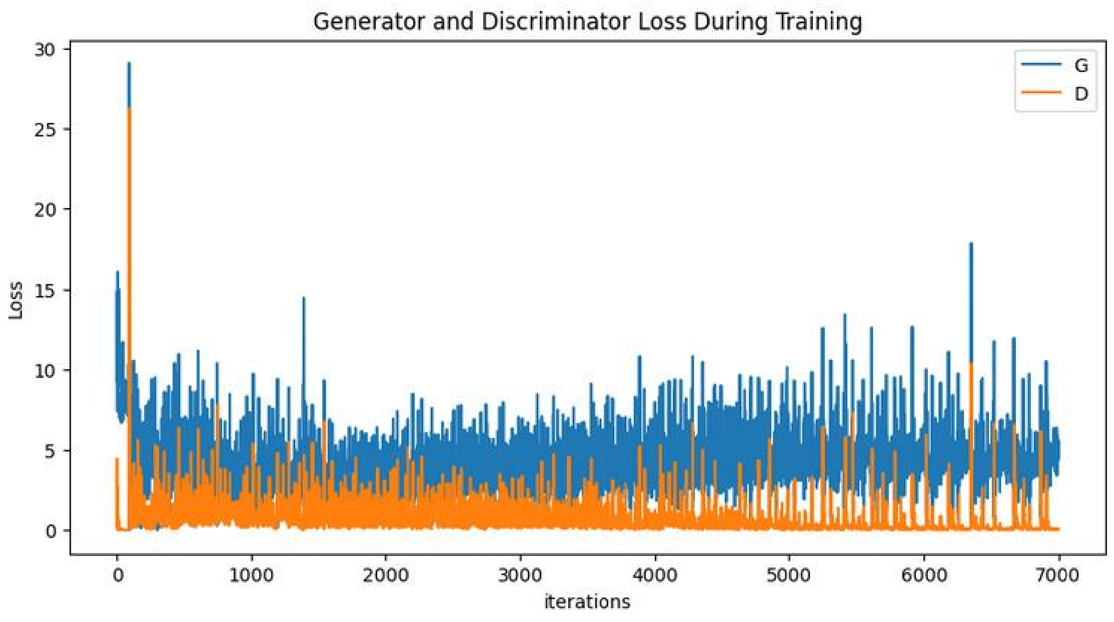


Figure 4.9 *: Generator and Discriminator Loss Graph*

Overall, analyzing the loss graph provides valuable insights into the training dynamics and performance of the GAN, helping to diagnose issues such as mode collapse, convergence problems, or instability, and guiding adjustments to training parameters or architectures to improve GAN performance.

We compare the real images with the generated images and on the basis of the fidelity score we make changes in the next epoch.

To calculate the next epoch after comparing a real and generated image based on fidelity score, we adjust the parameters of our training algorithm. Here's a general approach:

Evaluate Fidelity Score: After each epoch, evaluate the fidelity score between the real and generated images.

Compare with Previous Epoch: Compare the fidelity score of the current epoch with the fidelity score of the previous epoch.

Adjust Training Parameters: If the fidelity score of the current epoch is higher (indicating better performance) than the previous epoch, we consider adjusting our training parameters to encourage the model to converge to a lower fidelity score.

Decrease Learning Rate or Change Model Architecture: Options for adjusting training parameters include decreasing the learning rate, changing the optimizer, or modifying the model architecture to encourage convergence to a lower fidelity score.

Repeat Training: Continue training with the adjusted parameters for the next epoch.

By continuously monitoring and adjusting the training parameters based on fidelity score comparisons, we aim to achieve better fidelity between the generated and real images over time.

Chapter 5: RESULTS AND DISCUSSION

After training our GAN model generates the following images as shown in Figure 5.1 which are further enhanced through super resolution GAN.

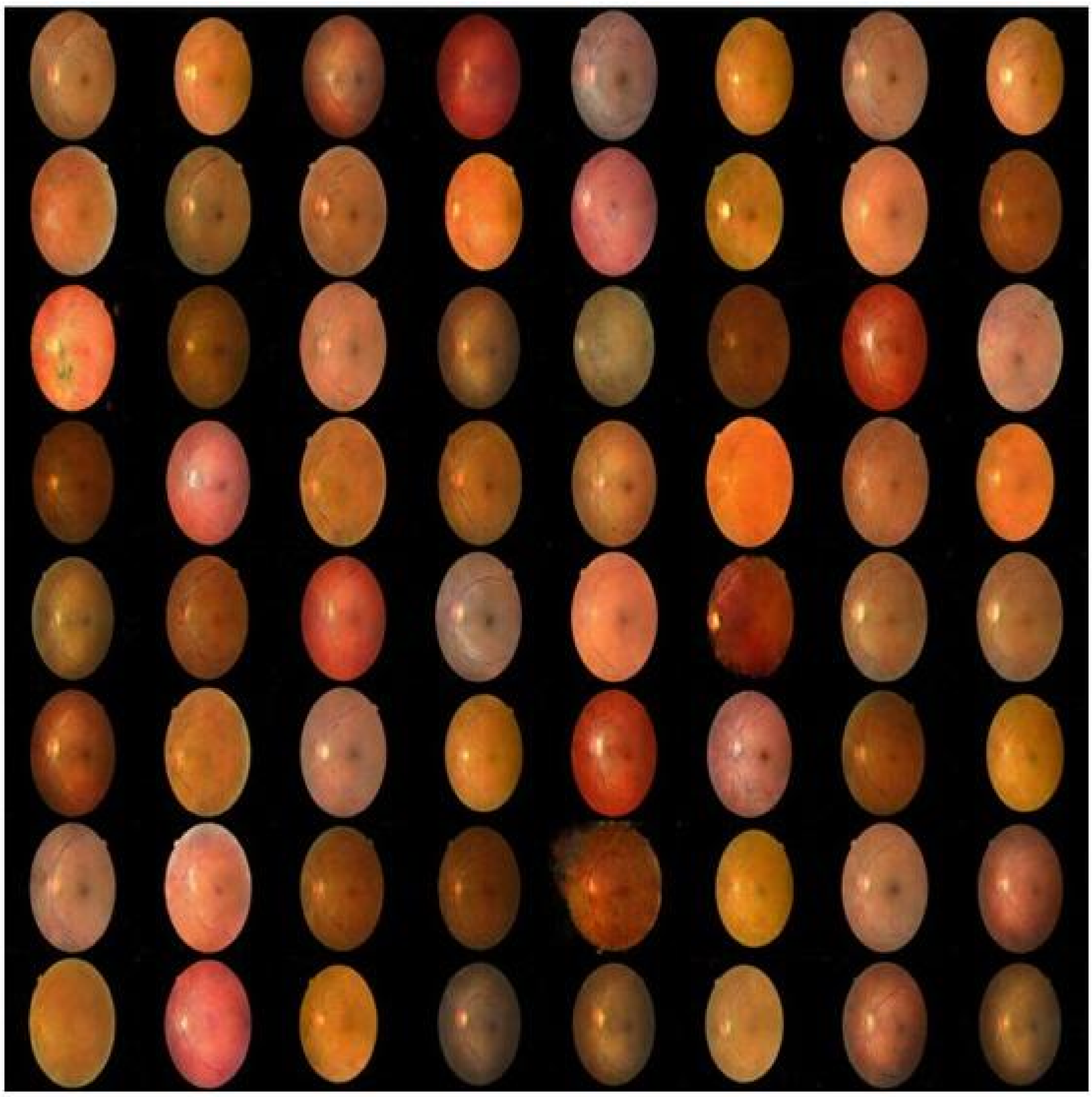
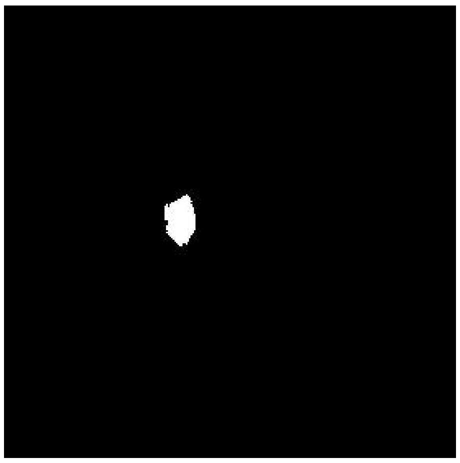
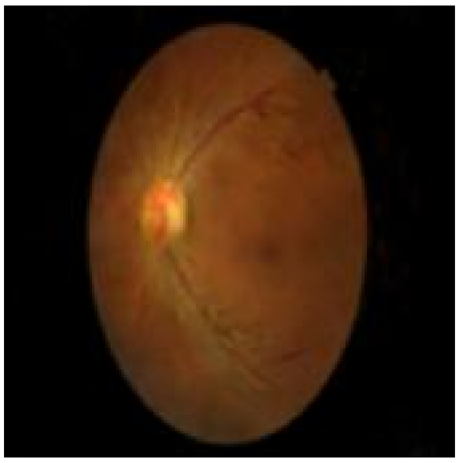


Figure 5.1 *: Generated Images*

Let's have a closer look at a generated image and its corresponding optic disc image in figure 5.2 without super-resolution.



(a)



(b)

Figure 5.2: *(a) Optic Disc Image*
(b) Generated Image

The above generated image after super-resolution is displayed in figure 5.3 below.

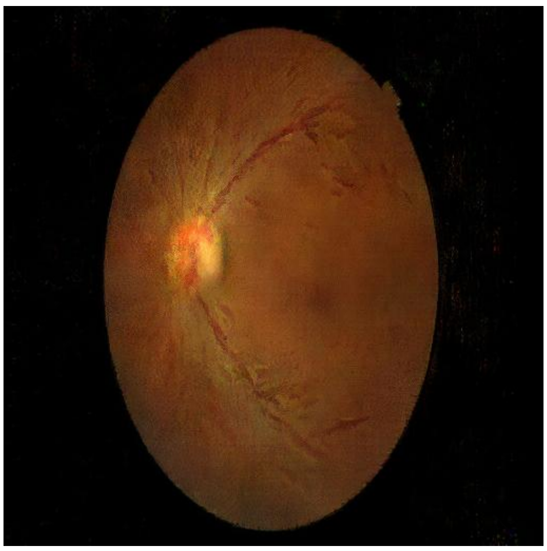


Figure 5.3 : *Generated Super-Resolution Image*

Chapter 6: CONCLUSION

In conclusion, the project represents a notable endeavor in harnessing the power of Generative Adversarial Networks (GANs) to generate synthetic retinal fundus images. By employing cutting-edge deep learning techniques, particularly leveraging the PyTorch framework, the project successfully trains a GAN to understand the intricate distribution of real fundus images and produce convincing synthetic counterparts.

Throughout the project, meticulous attention is paid to every aspect, from the design of the generator and discriminator networks to the optimization of training parameters. The generator network adeptly learns to transform random noise vectors into realistic fundus images, while the discriminator network becomes proficient in distinguishing between real and synthetic images. This adversarial training mechanism drives iterative improvements in both networks, resulting in the generation of increasingly authentic synthetic images.

The training process is executed with precision, employing the Binary Cross-Entropy (BCE) loss function and Adam optimizers to guide network optimization effectively. Training progresses through numerous epochs, with regular evaluation of discriminator and generator losses to monitor convergence and ensure model stability. Additionally, safeguards are implemented to halt training if specific criteria are met, thus preventing overfitting and model degradation.

Furthermore, the project includes provisions for visualizing the generated images, allowing for qualitative assessment of the network's performance. By periodically saving trained models, the project facilitates reproducibility and future experimentation, laying the groundwork for further advancements in the field.

In this project we separate the original retinal image into three BGR channels and create an image with excluding the blood vessels. With our aim to fill the blood vessels we fill up the blood vessels using a pre-defined logic of an 11x11 matrix window to create a new green image. We narrow down the matrix window and apply K-means clustering algorithm on the resultant new green image. After feature calculation we create three clusters and generate three images having high contrast region, low contrast region and an optic disc region. The region with high intensity is the high contrast region and the region with low intensity is the low contrast region.

After this procedure we generate optic disk regions and then train our GAN model by passing optic region dataset to the generator and the fundus image dataset to the discriminator. The generated images are identified by the discriminator which gives them a real or fake pass.

In essence, this project represents a significant stride towards the integration of generative models into retinal funduscopy, with promising implications for medical imaging, education, and research. While additional refinement and validation may be necessary for clinical deployment, the project underscores the potential of GANs to augment diagnostic capabilities and enhance patient care in ophthalmology and beyond. Through ongoing innovation and collaboration, the integration of generative models holds the potential to revolutionize medical imaging workflows and drive meaningful advancements in healthcare.

Chapter 7: FUTURE SCOPE

A major goal for us is the generation of all the three regions and train our GAN model to further enhance the imaging aspect of the retinal fundus images.

In future endeavors, the integration of high contrast and low contrast regions alongside the optic disk region represents a promising avenue for enhancing the quality and diversity of synthetic retinal fundus images generated by the GAN. By incorporating additional regions of interest and varying contrast levels, we can provide the GAN with a more comprehensive understanding of retinal anatomy and pathology, ultimately improving its ability to generate realistic and clinically relevant images.

REFERENCES

The references used throughout our minor project are as follows:

[1]: AndrewMvd. Drive Digital Retinal Images for Vessel Extraction. [Dataset]. Kaggle. Available from: https://www.kaggle.com/datasets/andrewmvd/drive-digital-retinal-images-for-vessel-extraction

[2]: Abbadi, N. K. E. & Saadi, E. H. A. (2013). BLOOD VESSELS EXTRACTION USING MATHEMATICAL MORPHOLOGY. Journal of Computer Science, 9(10), 1389-1395. https://doi.org/10.3844/jcssp.2013.1389.1395

[3]: D.Marín, A. Aquino, M. E. Gegundez-Arias and J. M. Bravo, "A New Supervised Method for Blood Vessel Segmentation in Retinal Images by Using Gray-Level and Moment Invariants-Based Features," in IEEE Transactions on Medical Imaging, vol. 30, no. 1, pp. 146-158, Jan. 2011, doi: 10.1109/TMI.2010.206433.

[4]: S. Gerasimou, H. F. Eniser, A. Sen and A. Cakan, "Importance-Driven Deep Learning System Testing," 2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Seoul, Korea (South), 2020, pp. 322-323.

[5]: Asloob Ahmad Mudassar, Saira Butt, "Extraction of Blood Vessels in Retinal Images Using Four Different Techniques", Journal of Medical Engineering, vol. 2013, Article ID 408120, 21 pages, 2013. https://doi.org/10.1155/2013/408120

[6]: Sundaram, Ramakrishnan & Ravichandran, K S & Jayaraman, Premaladha & B, Venkatraman. (2019). Extraction of Blood Vessels in Fundus Images of Retina through Hybrid Segmentation Approach. Mathematics. 7. 169. 10.3390/math7020169.

[7]: Chanwimaluang, Thitiporn & Fan, Guoliang. (2003). An efficient algorithm for extraction of anatomical structures in retinal images. Proceedings / ICIP ... International Conference on Image Processing. 1. 1093-1096. 10.1109/ICIP.2003.1247157.

[8]: Dong C. Image semantic segmentation method based on GAN network and ERFNet model. J Eng. 2021; 2021: 189–200. https://doi.org/10.1049/tje2.12025

[9]: Radford, A., Metz, L., & Chintala, S. (2015, November 19). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv.org. https://arxiv.org/abs/1511.06434

[10]: ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. (n.d.). https://esrgan.readthedocs.io/en/latest/