ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

Medical Image Synthesis using Generative Adversarial Network

Authors

Antara Risha, 180042146

Shaira Saiyara Islam, 180042147

Anika Tahsin, 180042150

Co-Supervisor

Tasnim Ahmed

Lecturer

Dept. of CSE, IUT

Supervisor

Tareque Mohmud Chowdhury

Assistant Professor

Dept. of CSE, IUT

A thesis submitted in partial fulfilment of the requirements

for the degree of B. Sc. in Software Engineering

Academic Year: 2021-2022

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

A Subsidiary Organ of the Organization of Islamic Cooperation (OIC)

Dhaka, Bangladesh

June 4, 2023

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out under the supervision of Tareque Mohmud Chowdhury, Assistant Professor of the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Antara Risha

____

Student ID - 180042146

Shaira Saiyara Islam

____

Student ID - 180042147

Anika Tahsin

____

Student ID - 180042150

Approved By:

Co-Supervisor:

Tasnim Ahmed

Lecturer

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Supervisor:

Tareque Mohmud Chowdhury

Assistant Professor

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Acknowledgement

We would like to express our grateful appreciation for **Tareque Mohmud Chowdhury**, Assistant Professor, Department of Computer Science and Engineering, IUT for being our adviser and mentor. His motivation, suggestions and insights for this research have been invaluable. Without his support and proper guidance this research would never have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to him.

We are also grateful to **Tasnim Ahmed**, Lecturer, Department of Computer Science and Engineering, IUT for his valuable inspection and suggestions on our proposal of Medical Image Synthesis.

Contents

Abstract

Medical image synthesis has emerged as a promising technique in the field of healthcare, enabling the generation of realistic medical images for various applications. This study focuses on medical image synthesis using Generative Adversarial Networks (GANs) applied to the IDRID dataset, which contains retinal images for diabetic retinopathy analysis. The objective of this research is to explore the potential of GANs in generating synthetic retinal images that closely resemble real patient data. The IDRID dataset provides a valuable resource for training and evaluating the GAN model. By leveraging the power of GANs, the proposed framework aims to generate high-quality synthetic retinal images with similar characteristics and visual appearance to real patient images. This has the potential to augment the existing dataset, expand its diversity, and improve the performance of diagnostic and treatment algorithms. The methodology involves training a GAN architecture consisting of a generator and a discriminator network. The generator network learns to generate synthetic retinal images from random noise, while the discriminator network evaluates the authenticity of the generated images. The two networks engage in an adversarial training process, where the generator aims to fool the discriminator into classifying the synthetic images as real. Evaluation of the synthesized retinal images includes quantitative metrics such as structural similarity index (SSIM), peak signal-to-noise ratio (PSNR), and analysis to assess the similarity and quality of the generated images compared to real IDRID dataset images. The outcomes of this research provide insights into the capabilities of GANs in generating realistic retinal images from the IDRID dataset. The generated images have the potential to enhance the limited availability of labeled medical data, facilitate algorithm development, and support computer-aided diagnosis systems. The findings contribute to the broader field of medical image synthesis, showcasing the potential of GANs in improving healthcare outcomes through enhanced image data availability and diversity.

1 Introduction

1.1 Overview

One use for the “Generative Adversarial Networks” (GANs), a deep learning model is medical image processing [1]. A generator and a discriminator are the two neural networks that make up GANs. The discriminator is trained to discriminate between real and synthetic data, and the generator is trained to make synthetic data that is comparable to a given set of real data [2].

Generating artificial medical images to supplement the training data for other machine-learning models is one potential application of GANs in the analysis of medical images [3]. This can be especially helpful when there is a shortage of real training data. GANs can help other machine learning models that are employed for tasks like diagnosis or prognosis perform better by producing synthetic images that are comparable to actual ones.

Enhancing the quality of medical photographs is another potential application of GANs in image processing [4]. GANs could be used, for instance, to improve the resolution of low-resolution photos or to eliminate noise or artifacts from medical imaging.

GANs have the potential to greatly increase the accuracy and efficiency of numerous machine-learning tasks using medical pictures, making them a promising tool in the field of medical image processing.

1.2 Motivation and Problem Statement

The following is a more general description of some of the issues that researchers studying medical image processing and synthesis have to deal with:

1. Lack of public datasets and insufficient data in public datasets: It can be difficult for researchers and clinicians to use machine learning techniques for tasks like diagnosis, prognosis, and treatment planning because there is

frequently a lack of public medical image datasets. There are many causes for this shortage, including:

(a) Privacy issues: It can be challenging to make medical photographs publicly available without breaking privacy rules since they frequently include sensitive personal information, such as names and identifying characteristics.

(b) A restricted supply: Due to the fact that they are often stored in PACS (Picture Archiving and Communication Systems), medical images are frequently only accessible for a short time. This means that it can be difficult to obtain a large, diverse dataset of medical images.

(c) High costs: Obtaining medical imaging can be quite expensive, especially for academics with tight budgets.

(d) Limited access: Access to medical imaging is frequently restricted to authorized people, which can make it challenging for researchers to get the information they require.

The shortage of public medical image datasets is a significant challenge in the field of medical image analysis and machine learning, and it limits the ability of researchers to advance our understanding of various medical conditions and improve patient care.

2. Imbalanced Dataset:

   Medical image datasets that are unbalanced do not have an equal amount of photos from each class. For instance, if there are much more photographs of non-cancerous tumors than cancerous ones, a dataset of medical images of cancerous and non-cancerous tumors may be unbalanced.

   When developing machine learning models, imbalanced datasets can be problematic since they can produce biased and subpar results. The reason for this is because machine learning algorithms are made to maximize the model's overall accuracy, which can be challenging to do when the data is unbalanced. Particularly, models developed from unbalanced datasets may perform better for classes that make up the majority while performing worse for classes that make up the minority.

1.3 Proposed Idea

Our main idea is to create a robust Conditional Generative Adversarial Network (cGAN) so that we can generate a sufficient amount of realistic-looking medical image data. This approach aims to address the imbalance between malignant and benign cases, thereby improving the accuracy of disease prediction models.

To accomplish this task, we will utilize one skin-related datasets: the Diabetic Retinopathy-related dataset, INDIAN DIABETIC RETINOPATHY IMAGE DATASET (IDRID) [5].

By leveraging this dataset, we aim to augment the existing data by generating synthetic images that closely resemble real medical images. This augmentation will effectively increase the size of the dataset, allowing for a more balanced representation of malignant and benign cases. Ultimately, this expanded dataset will contribute to training more accurate disease prediction models.

1.3.1 Diabetic Retinopathy

Diabetic retinopathy is a type of eye damage that can occur in people with diabetes. It is caused by changes in the blood vessels of the retina, the light-sensitive layer of tissue at the back of the eye.

In people with diabetic retinopathy, the blood vessels in the retina may become damaged and leak fluid or blood into the eye. This can cause vision loss or blindness if left untreated.

Diabetic retinopathy is a serious complication of diabetes and is a leading cause of vision loss in adults. It is more likely to occur in people with poorly controlled diabetes and in those who have had diabetes for a long time.

The risk of developing diabetic retinopathy can be reduced by maintaining good blood sugar control, getting regular eye exams, and managing other risk factors for diabetes, such as high blood pressure and high cholesterol. Treatment for diabetic retinopathy may include laser surgery, injections of medications into the eye, or surgery to remove the damaged blood vessels.

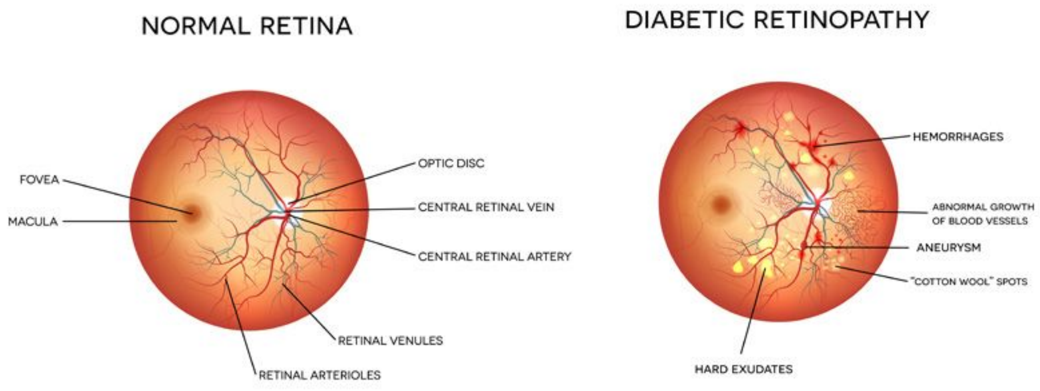


Figure 1: A normal and diabetic retina.

The INDIAN DIABETIC RETINOPATHY IMAGE DATASET (IDRID) [5] is a collection of medical images of the retina that was created to support the development of machine learning algorithms for the diagnosis of diabetic retinopathy. The IDRID dataset contains images of the retina from individuals with and without diabetic retinopathy, as well as images of various stages of the disease.

The IDRID dataset was created by a team of researchers in India as part of a larger effort to improve the diagnosis and management of diabetic retinopathy. The dataset is intended to be a resource for the development and evaluation of machine learning algorithms for the diagnosis of diabetic retinopathy, and it is widely used in the field of medical image analysis.

The IDRID Dataset is available for download by researchers and clinicians who are interested in using it for machine learning or other research purposes.

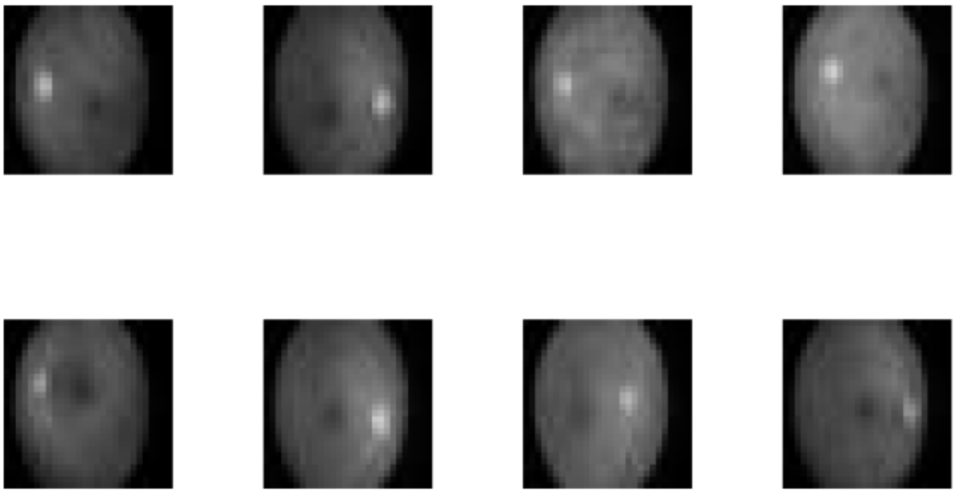


Figure 2: Some data samples from IDRID Dataset

1.4 Research Challenges

1. **Reflecting Real Data Accurately:** Making sure the output accurately reflects actual patient data is one of the problems of working with medical data. In order to accomplish this, the network must be trained on a comprehensive and representative dataset that includes information on a variety of medical disorders, patient demographics, and imaging techniques. The generated images may not accurately depict the underlying medical issues or fail to offer insightful information if the complexity and unpredictability of real data are not captured.

2. **Maintaining Non-Traceability of Medical Data:** Medical records contain sensitive and private information about patients' personal health. It is imperative to make sure that the created samples of synthetic medical data do not reveal any recognizable or traceable information about actual patients. In order to safeguard patient confidentiality, it is imperative to find a balance between gathering accurate data and protecting privacy.

3. **Variation in Terms of Image Acquisition Set-Up, Colors, and Sizes:** In terms of acquisition characteristics, such as imaging machines, protocols,

and settings, medical imaging data might show wide differences. Images can also differ in terms of colors, resolutions, and sizes. The generative model must be trained to accommodate these variances and provide images that are consistent with various acquisition setups. The usefulness and usability of the generated data are limited since failing to take these variances into account can result in unrealistic or inconsistent generated images.

4. **Biased Output of the Network:** The biases found in the training data can affect generative models. The resulting output may potentially show biases if the training dataset is biased toward particular demographics, medical problems, or imaging modalities. Due to erroneous portrayals of minority communities or illnesses, inequities in medical research and application may result. The careful curation and augmentation of the training dataset, as well as the creation of training methods that are fairness-aware, are necessary to address and mitigate bias in generated data.

5. **Evaluation and Validation of Generated Data:** A crucial problem is determining the validity and dependability of generated medical data. It's possible that conventional computer vision evaluation measures like structural or pixel-level similarity are insufficient to assess the clinical utility and accuracy of the generated images. To assure the generated data relevance and trustworthiness in medical research, diagnosis, and treatment planning, it is essential to develop rigorous evaluation procedures that include expert knowledge and clinical validation.

6. **Ethical Considerations:** The creation of synthetic medical data raises ethical questions about possible misuse or incorrect interpretation of the produced samples. When developing and deploying generative models in the medical field, it is crucial to ensure the appropriate and ethical use of produced data, set rules for data sharing, and address any potential legal or privacy consequences. To preserve confidence and integrity in the collection of medical data, it is crucial to properly handle issues including patient rights

protection, informed consent, and data governance.

1.5 Organization of Thesis

The topic and the range of our research are succinctly described in the introduction section. It clarifies the rationale behind deciding to concentrate our study on general adversarial network (GAN) [1] and medical image synthesis.

We go into the background research and literature evaluation required for our thesis research in the second chapter. This section includes a thorough analysis of pertinent publications and datasets. It provides the framework for our investigation, guiding how we interpret the body of knowledge and guiding the design of our architectural framework.

The proposed methodology section describes our research process and the architecture we used to get the findings we sought. It guarantees transparency and replicability by providing a thorough justification of our methodology.

A comparative analysis is done to compare our results with existing architecture results from other literature studies, validating the efficacy of our methodology. This analysis highlights our work's contributions and accomplishments while serving as a baseline for future effort.

Finally, an assessment is made in light of our findings and outcomes. We highlight the most important findings from our research and analyze the ramifications and importance of each. We also point out potential directions for future research and offer ways to improve and broaden the scope of our study.

The thesis report encapsulates the collective efforts of our thesis team, guided by our supervisor and co-supervisor. It provides a comprehensive account of our research journey, from the initial motivation to the final conclusions and future prospects.

2 Background Study

2.1 Medical Image Modality

A medical image modality [6] is the particular imaging method or piece of equipment utilized to create the image in the context of medical imaging. Each modality gives distinct information for diagnosis and therapy and captures various facets of the human body, such as anatomy, function, or metabolism. Common medical image modalities include:

1. **X-ray**: Ionizing radiation is used in X-ray imaging to provide images of bones and some soft tissues. It is frequently used to diagnose tooth issues, lung ailments, and fractures.

2. **Computed Tomography (CT)**: CT scans create fine cross-sectional images of the body using X-rays. Visualizing interior structures including organs, blood arteries, and malignancies is one of the main uses of CT scans.

3. **Magnetic Resonance Imaging (MRI)**: Strong magnetic fields and radio waves are used in magnetic resonance imaging, or MRI, to provide precise images of soft tissues and organs. It is frequently employed to assess the spine, joints, abdomen, and brain.

4. **Ultrasound**: Using high-frequency sound waves, ultrasound imaging can produce real-time images of organs, blood vessels, and tissues. It is frequently used to check the heart, abdomen, and blood flow, and track pregnancies.

5. **Positron Emission Tomography (PET)**: A radioactive tracer that releases positrons is injected into the patient's body during a PET scan. Images that show metabolic activity in tissues are produced by the detection of the released gamma rays. PET is useful for identifying cancer and evaluating brain function.

6. **Single-Photon Emission Computed Tomography (SPECT)**: Similar to PET, SPECT imaging employs radioactive tracers but uses different angles to catch gamma rays. It is frequently employed for bone, cardiovascular, and neurological imaging.

2.2 Medical Image Synthesis

The method of creating artificial medical images using computer algorithms is known as medical image synthesis [3]. These artificial images, which are created to resemble actual medical images, can be employed for a number of tasks, such as data augmentation, image quality enhancement, and model training.

Medical image synthesis is frequently used to produce more training data for machine learning models. Synthetic images can be used to augment the data and boost the model's performance when the amount of genuine training data is constrained. Testing the resilience and generalization of machine-learning models can also be done using synthetic images.

Medical image synthesis can be achieved using a variety of techniques, including Generative Adversarial Networks (GANs) [7], which are a type of deep learning model that can be used to generate synthetic data that is similar to a given set of real data. Other techniques for medical image synthesis include image translation, image-to-image translation, and image super-resolution.

Medical image synthesis is a useful tool for improving the accuracy and efficiency of machine-learning models that are used for tasks related to medical images.

2.3 Importance of medical data synthesis

Medical data synthesis is the process of combining and evaluating multiple sources of data to create a comprehensive overview of a particular medical topic or issue. This can be important for a number of reasons [3] :

1. Medical data synthesis allows researchers to draw more robust conclusions from their data. By combining multiple studies, researchers can increase the

sample size and statistical power of their analysis, which can lead to more reliable findings.

2. Synthesizing data from multiple sources can help identify patterns and trends that might not be apparent from a single study. This can be especially useful for identifying trends over time or in different populations.

3. Medical data synthesis can help to identify gaps in the existing research on a particular topic. By reviewing all of the available data on a topic, researchers can identify areas where more research is needed.

4. Synthesizing data can be helpful for making clinical decisions or developing treatment guidelines. By reviewing all of the available evidence on a particular topic, clinicians can make informed decisions about how to care for their patients.

A medical data synthesis is an important tool for advancing our understanding of various medical topics and for improving patient care.

2.4 Generative Adversarial Network (GAN)

A Generative Adversarial Network (GAN) [1] is a type of deep learning model that is used to generate synthetic data that is similar to a given set of real data. It consists of two neural networks: a generator and a discriminator. The generator is trained to produce synthetic data that is similar to the real data, while the discriminator is trained to distinguish between real and synthetic data.

The training process for a GAN involves a competition between the generator and discriminator. The generator produces synthetic data, and the discriminator tries to identify whether the data is real or synthetic. The generator receives feedback on its performance from the discriminator, and it uses this feedback to improve its ability to generate realistic synthetic data. The discriminator, in turn, becomes better at identifying synthetic data. This process continues until the generator is able to produce synthetic data that is indistinguishable from real data, and the discriminator is unable to distinguish between the two.

GANs have been used for a variety of tasks, including image generation, text generation, and anomaly detection. They have also been applied to medical image processing, where they can be used to generate synthetic medical images or to improve the quality of real medical images.

2.4.1 Importance of using GAN in medical image synthesis

There are several potential benefits of using Generative Adversarial Networks (GANs) for medical image synthesis [3] :

1. Data augmentation: One potential use of GANs in medical image synthesis is to generate synthetic medical images that can be used to augment the available training data for other machine learning models. This can be especially useful in situations where the amount of real training data is limited. By generating synthetic images that are similar to real ones, GANs can help improve the performance of other machine learning models that are used for tasks such as diagnosis or prognosis.

2. Improved image quality: GANs can also be used to improve the quality of real medical images. For example, they could be used to remove noise or artifacts from images or to enhance the resolution of low-resolution images.

3. Increased efficiency: Using GANs to generate synthetic medical images can be more efficient than manually collecting and labeling large amounts of real data. This can be especially useful in situations where it is difficult or time-consuming to obtain real data.

4. Greater flexibility: GANs can be used to generate synthetic images that are diverse and varied, which can be useful for training machine learning models to be more robust and generalizable.

GANs have the potential to significantly improve the accuracy and efficiency of various machine-learning tasks related to medical images.

2.4.2 Conditional Generative Adversarial Network (cGAN)

A Conditional Generative Adversarial Network (cGAN) [4] is a type of Generative Adversarial Network (GAN) that is used to generate synthetic data that is conditioned on a given set of input data. Like a regular GAN, a cGAN consists of two neural networks: a generator and a discriminator. The generator is trained to produce synthetic data that is similar to the real data, while the discriminator is trained to distinguish between real and synthetic data.

The key difference between a regular GAN and a cGAN is that the generator in a cGAN is conditioned on a set of input data, which means that it is able to generate synthetic data that is related to the input data in some way. For example, a cGAN could be used to generate synthetic images of faces that are conditioned on a set of input images of faces, in which case the generator would be able to generate synthetic images that are similar to the input images.

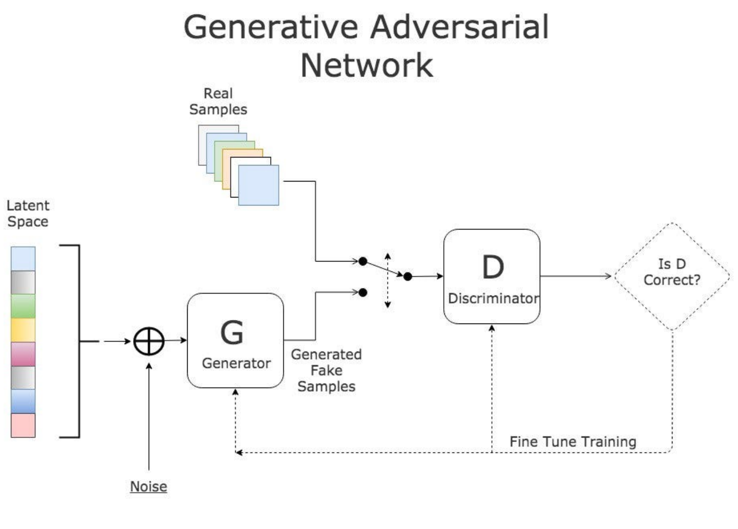


Figure 3: *The architectural flow of a conventional cGAN model.*

cGANs have been used for a variety of tasks, including image generation, text generation, and anomaly detection. They have also been applied to medical image processing, where they can be used to generate synthetic medical images or to improve the quality of real medical images.

2.4.3 Benefits of Using cGAN instead of Vanilla GAN

There are several reasons why a Conditional Generative Adversarial Network (cGAN) might be preferred over a vanilla Generative Adversarial Network (GAN) for medical image synthesis [4] :

1. Greater control over the synthesis process: With a cGAN, it is possible to specify certain conditions or constraints that the synthetic images must satisfy. This can be useful in situations where it is important to ensure that the synthetic images are similar to the real ones in certain ways. For example, a cGAN could be used to generate synthetic images of skin lesions that are conditioned on the type of lesion (e.g. melanoma, basal cell carcinoma, etc.), which would allow the generator to produce synthetic images that are more representative of the real ones.

2. Improved synthesis quality: By conditioning the generator on a set of input data, it is possible to improve the quality of the synthetic images that are generated. This can be especially useful in situations where the real data is noisy or of low quality, as the cGAN can use the input data to guide the synthesis process and produce more realistic synthetic images.

3. Greater flexibility: cGANs can be used to generate synthetic images that are diverse and varied, which can be useful for training machine learning models to be more robust and generalised.

Conclusively, cGANs offer greater control, improved synthesis quality, and greater flexibility compared to vanilla GANs, which makes them a useful.

2.4.4 Deep Convolutional Generative Adversarial Network (DCGAN)

DCGAN [2] stands for Deep Convolutional Generative Adversarial Network. It is a type of generative model that uses a combination of deep convolutional neural networks (CNNs) and adversarial training to generate realistic synthetic data, such as images.

DCGANs are based on the framework of Generative Adversarial Networks (GANs), which consist of two main components: a generator network and a discriminator network. The generator network takes random noise as input and learns to generate synthetic data samples that resemble the training data. The discriminator network, on the other hand, learns to distinguish between real and fake data samples.

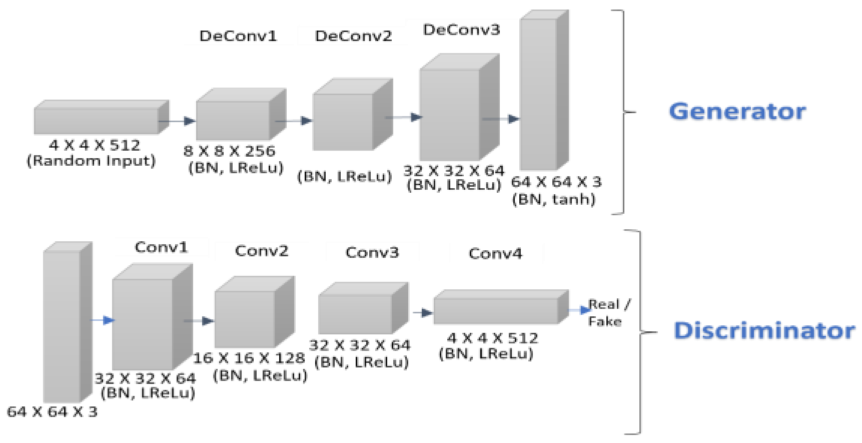


Figure 4: *The generator and discriminator architecture of a conventional DCGAN model [8].*

In DCGAN, both the generator and discriminator networks are designed using deep convolutional layers. The generator network typically starts with a low-resolution input and gradually upsamples the data using transpose convolutions to generate higher-resolution outputs. This allows the generator to capture spatial dependencies and generate more realistic images.

The discriminator network, on the other hand, uses a series of convolutional layers to learn features from both real and generated images. It learns to classify whether an input image is real or fake. The goal of the discriminator is to correctly classify real images as real and generated images as fake, while the generator aims to generate images that can fool the discriminator.

DCGANs are trained in an adversarial manner, where the generator and discriminator networks are trained simultaneously. The training process involves iteratively updating the networks using backpropagation and gradient descent. The generator tries to minimize the discriminator's ability to distinguish between real and fake samples, while the discriminator tries to maximize its discriminative accuracy.

DCGANs have been successfully applied to various image generation tasks, such as generating realistic human faces, natural scenes, and medical images. They have contributed to significant advancements in the field of generative models and have opened up new possibilities for creating high-quality synthetic data.

2.4.5 Benefits of DCGAN in Medical Image Synthesis

DCGAN (Deep Convolutional Generative Adversarial Network) is a type of generative model that has shown promise in various domains, including medical image synthesis. Here are a few reasons why DCGAN is often preferred for medical image synthesis:

1. Detecting spatial features: X-rays and MRI scans of medical objects frequently show complicated spatial trends and structures. DCGAN's deep

convolutional architecture allows it to be effective in capturing spatial features. Intricate details found in medical images can be modeled by DCGAN through the convolutional layers' ability to develop hierarchical representations of images.

2. Generative capacity: DCGAN was created primarily to produce realistic images of the highest quality. DCGAN is able to understand the underlying distribution of medical images in the context of medical image synthesis and produce new images that closely resemble the original data. Applications like data augmentation, anomaly detection, or creating artificial datasets for training other models can all benefit from this.

3. Unsupervised learning: DCGAN is capable of learning to create medical images without the need of explicit labels or annotations during the training phase. This can be helpful in the field of medical imaging, where it can be difficult and time-consuming to collect big labeled datasets. Without depending on labeled samples, DCGAN may learn from unannotated medical images and produce synthetic data by utilizing unsupervised learning.

4. Training under adversarial conditions: DCGAN combines a generative and discriminative framework, simultaneously training a generator network and a discriminator network. The discriminator gains the ability to tell real images from generated ones, while the generator gains the ability to create more realistic images. The generator's ability to synthesize images gets better over time thanks to this adversarial training process. DCGAN can create visually appealing and clinically believable medical images by utilizing the power of adversarial training.

Though DCGAN has demonstrated potential in the synthesis of medical images, it is crucial to bear in mind that the generated images need to be rigorously verified and evaluated for their clinical relevance and accuracy.

2.5 Literature Review

2.5.1 GANs for Medical Image Synthesis

Powerful Generative Adversarial Networks (GANs) now produce astounding photorealistic images that mirror the contents of datasets they were trained to duplicate. The question of whether GANs can be as effective at producing usable medical data as they are at producing realistic RGB images is a recurring one in the field of medical imaging. A recent study by Skandarani et al in their paper "GANs for Medical Image Synthesis: An Empirical Study" concluded with a performance of a multi-GAN and multi-application study to gauge the benefits of GANs in medical imaging [1]. They experimented with different GAN architectures, ranging from the straightforward DCGAN to the more complex style-based GANs, on three different medical imaging modalities and organs: cardiac cine-MRI, liver CT, and RGB retina pictures. To gauge the visual acuity of the images produced by GANs, their FID score was estimated from training data taken from well-known and frequently used datasets. By assessing the segmentation accuracy of a U-Net trained on these produced images and the original data, their use was further evaluated. The result of the comparative analysis and study of different GAN architectures in generating fake images from realistic images gave the conclusion that the best GANs can produce medical images that look realistic by FID standards and that pass certain metrics. They can also deceive educated experts in a visual Turing test. But according to segmentation results, no GAN is able to replicate the complete richness of medical datasets.

2.5.2 GAN-based generative modelling for dermatological applications

The main objective behind selecting dermatological images of skin lesions was because of a rich dataset which is widely being used in different dermatological disease related researches. The dataset which is a public dataset is already sufficiently large but has limitations which hinder the performance of the trained models and architectures and also has an issue with the amount of data samples to support disease detection and classification into two classes -benign and malignant. The dataset we considered is the International Skin Imaging Collaboration Database,2020 (ISIC) which was used in a study by Limeros et al, Sandra Carrasco et al, Sylwia Majchrowska et al which produced the paper published in 2022 at a conference. The study “GAN-based generative modelling for dermatological applications – comparative study” investigated both centralized and decentralized unconditional and conditional GANs [9]. The decentralized environment resembles a more realistic hospital scenario with three institutions, while the centralized option mimics studies on large but extremely uneven skin lesion datasets. The models’ performance in terms of fidelity, diversity, speed of training, and predictive ability of classifiers trained on the generated synthetic data was also evaluated in this study. The limitations of the dataset used was profoundly visible. The dataset was highly unbalanced – only 2% of it’s samples belonged to histopathologically confirmed malignant melanoma cases. Also, there were significant variations in image acquisition set up of samples from patients in the lab results in bias of the dataset. These variations resulted in the bias in dataset which consequently affected the robustness of trained models.

The preparation and annotation of medical data is an expensive process that necessitates the help of medical professionals. Additionally, because of patient privacy concerns, access to medical data requires a crafted approval process. It becomes impossible for different institutions to share data and thus expertise with one another. On the other hand, if it is modeled correctly, synthetic data, which is data that is created entirely from scratch, cannot be linked to any specific person. Artificial data can be used in two different ways: first, to expand on limited, imbalanced datasets (such as those of rare diseases), and second, to anonymize data (to replace instead of augment real samples). In both cases, artificial medical data must achieve two opposing objectives. The data should provide the people whose records were used to build it with robust privacy protection while also correctly reflecting the original data.

The main contribution of this work was a detailed study of GAN-based artificial data generation in the case of one of the latest and largest open-source databases of skin lesions, namely, International Skin Imaging Collaboration (ISIC), 2020. Their research was based on StyleGAN2 with adaptive discriminator augmentation (ADA) architecture, which is considered the state-of-the-art in image generation.

There are several things to think about before using GANs to produce synthetic healthcare data. One should think about the architecture first. The efficacy of the selected designs primarily depends on computational resources and time; for example, unconditional GAN can be a viable option with a limited number of classes due to the lengthy training time of a single GAN. The GAN should be trained centrally if there is a large, annotated dataset available. Second, several viewpoints should be used to examine the generated synthetic data. The two characteristics that are frequently highlighted are fidelity and diversity, which are crucial in determining how accurately the synthetic data resembles the underlying real data. It is vital to check the veracity of the synthetic instances to ensure they are not merely replicating the training data, as the goal in healthcare is to prevent sharing data. The synthetic data should also be equally helpful as the real data for the future task (such as classification), and they should not permit inferences based on traits that are unrelated to the case, such as those connected to the data collection method.

2.5.3 Generating realistic Retinal Fundus Images for Diabetic Retinotherapy Classification using GAN based model

The second dataset in target for the generation of a synthetic dataset is INDIAN DIABETIC RETINOPATHY IMAGE DATASET (IDRID) which is an insufficiently small dataset with images not more than 300 in per class samples. It constitutes typical diabetic retinopathy lesions and also normal retinal structures annotated at a pixel level and provides information on the disease severity of diabetic retinopathy, and diabetic macular edema for each image. There are three parts of this dataset: segmentation, disease grading and localization. The images are classified into five classes from mild to severe based on the severity and dense ness of the images. But the prime limitation faced during the study conducted by Gupta Siddharth, Avnish Panwar, Silky Goel, Ankush Mittal, Rahul Nijhawan and Amit Kumar Singh. "Classification of Lesions in Retinal Fundus Images for Diabetic Retinopathy Using Transfer Learning." 2019 International Conference on Information Technology (ICIT) (2019) was the scarcity of data from this public dataset [5]. It has very less amount of training and testing data.

Their main goal was to classify the lesions in retinal color fundus images showing levels of severity in diabetic retinopathy using deep learning-based feature extraction and transfer learning-based approach using VGG19 network. In this study, the authors trained their DL model to identify the variety of lesions present in DR-affected retina with the goal of detecting and classifying the severity of DR. For the tasks of image embedding and classification, respectively, a pretrained deep convolution network VGG19 coupled with a number of classifiers was used. With a high degree of accuracy, their proposed model and classifiers classify the various lesions in DR. Based on the frequency of these 5 types of lesions, the severity of DR is divided into distinct classifications. In order to determine the severity of the DR in a certain image, a framework that can accurately differentiate between various types of lesions was also developed in this paper. They used the IDRID dataset, which contains 122 photos, of which 66% were used to train the model and the remaining 34% to test it. This amount of sample data is not satisfactory which is where our contribution comes. We will be generating realistic looking retinal fundus images from given small samples and increase the amount of training and testing data.

3 Methodology

The methodology using a cGAN (conditional Generative Adversarial Network) for medical image synthesis would typically involve the following steps:

1. Data preparation: The first step is to prepare the data that will be used to train the cGAN. This may involve pre-processing the images to ensure that they are in a suitable format and selecting a subset of the images to use as training data.

2. Model design: Next, the cGAN model must be designed. This involves specifying the architecture of the generator and discriminator networks and determining the appropriate loss functions and optimization algorithms.

3. Model training: Once the cGAN model has been designed, it can be trained on the prepared data. This typically involves feeding the images and their corresponding labels into the model and adjusting the model's parameters through back propagation and optimization.

4. Model evaluation: After the cGAN model has been trained, it can be evaluated to determine how well it performs at generating synthetic images. This may involve comparing the synthetic images to real ones or evaluating the performance of machine learning models that are trained on the synthetic images.

5. Model fine-tuning: If the cGAN model does not perform as well as desired, it may be necessary to fine-tune the model by adjusting its parameters or modifying its architecture.

6. Synthetic image generation: Once the cGAN model is performing well, it can be used to generate synthetic images by providing it with the desired labels and allowing it to generate the corresponding images.

The use of a cGAN for medical image synthesis involves preparing and labeling the data, designing and training the model, evaluating its performance, and fine-tuning it as needed.

3.1 Pipeline of our methodology

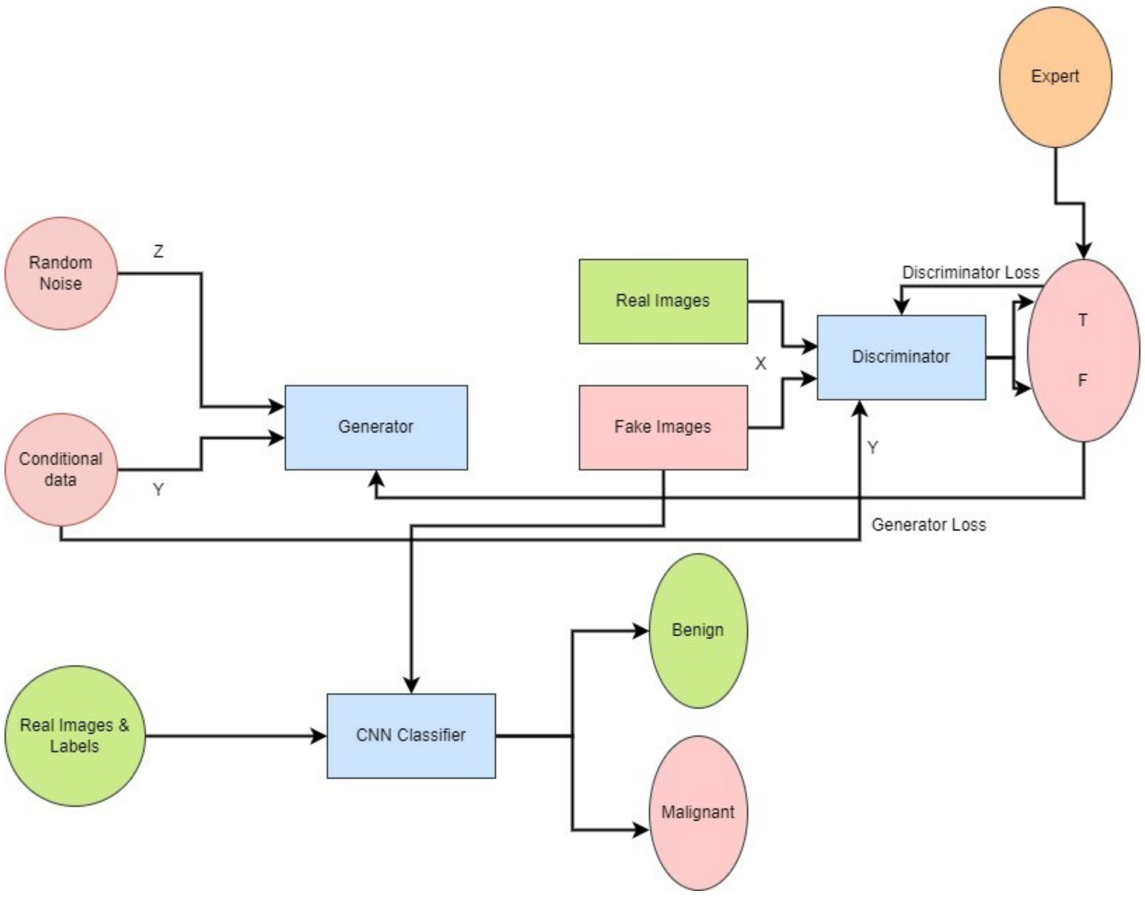


Figure 5: A flowchart depicting our proposed architecture.

3.2 Data Pre-processing

During the data preprocessing stage, our objective was to prepare the dataset for performing a comparison between the generated images and the original images. Initially, the dataset was not divided into classes, so we utilized Python to classify the dataset into five distinct classes based on the provided annotations in a CSV (Comma Separated Values) file, which was provided with the dataset and the images were annotated and the image annotation values were kept in the CSV files by medical personnel.

To begin the preprocessing, we loaded the original images along with their corresponding annotations from the CSV file. The CSV file contained relevant information such as file paths or names of the original images and the associated class labels. This allowed us to establish a connection between the images and their respective categories.

Next, we focused on resizing the original images to a standardized size of 28x28 pixels. Resizing the images was essential to ensure consistency in the dataset and facilitate the subsequent comparison process. To achieve this, we leveraged popular image processing libraries such as OpenCV or PIL (Python Imaging Library). These libraries offered convenient functions to resize the images while maintaining their original aspect ratio.

Furthermore, in order to create a suitable representation for the comparison task, we converted the resized images to grayscale. By converting the images to a single-channel representation (28x28x1), we effectively removed the color information while preserving the essential structural details. The grayscale conversion was again accomplished using libraries such as OpenCV or PIL.

To ensure uniformity and ease of processing, we proceeded to normalize the pixel values of the grayscale images. This involved scaling the pixel values to a suitable range, commonly between 0 and 1. By dividing the pixel values by 255, we achieved normalization, enabling better convergence during subsequent training or analysis.

Additionally, if required, we split the preprocessed dataset into training and testing sets. This division allowed us to utilize a portion of the data for training our models while reserving another portion for evaluating their performance. Proper dataset splitting is crucial to obtain reliable and unbiased model assessments.

Finally, for compatibility with deep learning frameworks such as TensorFlow or PyTorch, we optionally converted the preprocessed image and label data into the appropriate format. This involved converting the data into tensors or other suitable representations that could be directly fed into the deep learning models.

By following these preprocessing steps, we successfully classified the dataset into five classes based on the provided annotations, resized the original images to 28x28 pixels, converted them to grayscale, normalized the pixel values, and potentially split the dataset for training and testing. These preparatory measures allowed us to perform accurate comparisons between the generated images and the preprocessed original images, paving the way for further analysis and evaluation of our models.

3.3 Implemented Architectures

3.3.1 cGAN using IDRID dataset

The generator model for a conditional generative adversarial network (GAN) is defined by this architecture. The generator requires two inputs: a category label of size 1 and a latent vector of size latent_dim. Based on these inputs, the generator's objective is to produce a 32x32 RGB image. Latent_dim (dimensionality of the latent vector) and n_classes (number of classes for the category label; default value is 5) are the two arguments that the function define_generator accepts. A 1-dimensional input is what is meant by the label input (in_label). Using an embedding layer, the categorical label is included into a continuous vector representation (li). 50 is the embedding size. The embedded label is next routed through a thick layer to add more nodes and increase the number of dimensions for eventual concatenation.(8, 8, 1) is the output shape, with 1 being the chan-

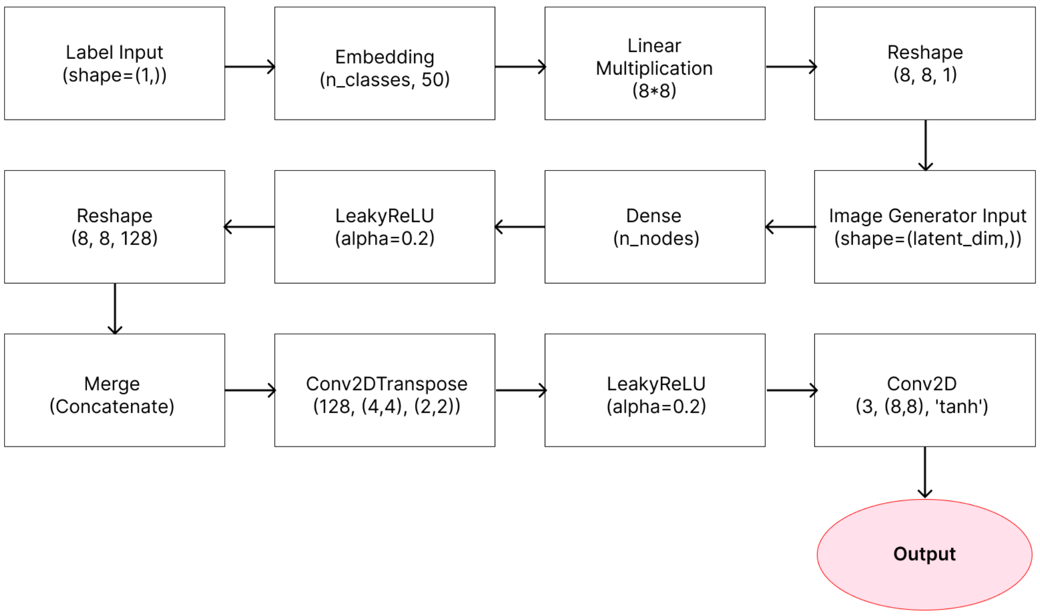

Label Input (shape=(1,))
Embedding (n_classes, 50)
Linear Multiplication (8*8)
Reshape (8, 8, 1)
Image Generator Input (shape=(latent_dim,))
Dense (n_nodes)
LeakyReLU (alpha=0.2)
Reshape (8, 8, 128)
Merge (Concatenate)
Conv2DTranspose (128, (4,4), (2,2))
LeakyReLU (alpha=0.2)
Conv2D (3, (8,8), 'tanh')
Output

Figure 6: A flowchart depicting the architecture of a generator model for cGAN using tensor flow.

nel dimension and 8x8 being the spatial dimension. The definition of the picture generator input (in_lat) is a latent vector input with size latent_dim. To add dimensions and modify the latent vector into a tensor of shape (8, 8, 128), a succession of dense and activation layers are applied. Along the channel dimension, the resulting image tensor and the label tensor are concatenated. After that, a transposed convolutional layer is applied to the concatenated tensor to upsample it to a size of 16x16 while keeping the number of channels constant. With the aid of a second transposed convolutional layer, the tensor is further upsampled to a size of 32x32. Finally, a convolutional layer with tanh activation is applied to the tensor to create an output image tensor with the shape (32, 32, 3) that represents an RGB image. The inputs (in_lat and in_label) and output layer (out_layer) are used to create the model, which is then returned.

The flowchart below defines the GAN (Generative Adversarial Network) model by connecting the generator (g_model) and discriminator (d_model) models.

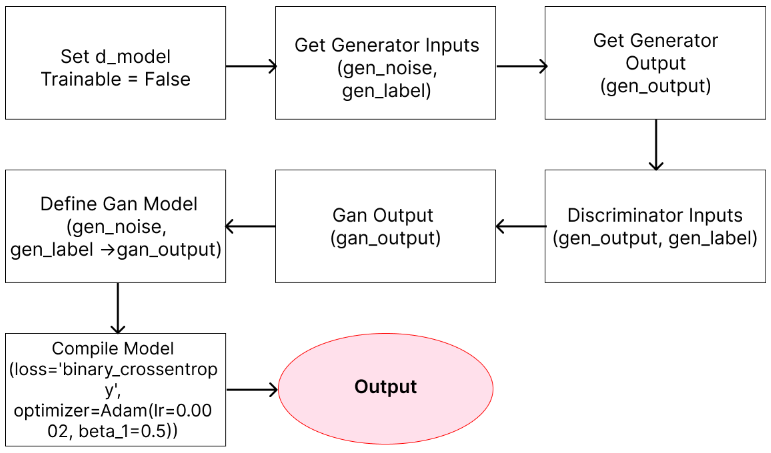


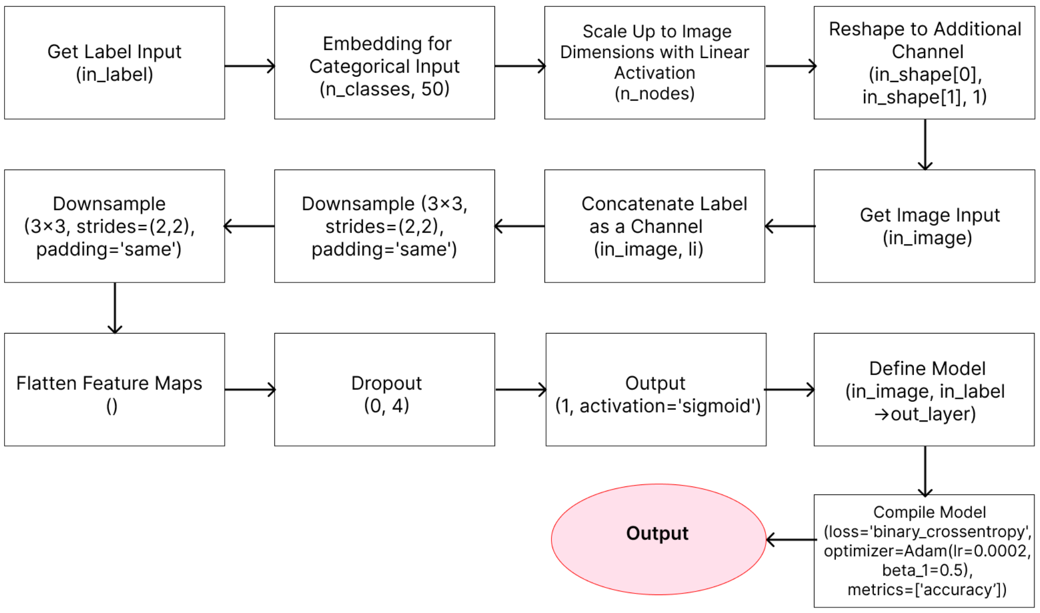
Figure 7: *The step wise depiction of features during definition of GAN model during cGAN model training using tensor flow.*

The trainable attribute of the discriminator model (d_model) has been set to False, making it untrainable. Because we wish to train the generator and discriminator models independently during GAN training, this is done. The noise vector gen_noise and the categorical label gen_label are the two inputs to the generator model. The generator model (g_model), from which these inputs are taken, is used. The created image is represented by an image tensor of the shape (32, 32, 3), which is the output of the generator model (gen_output).

The discriminator model (d_model) receives as inputs the output from the generator and the appropriate input label. The discriminator model assesses the created image and forecasts its veracity of it. The discriminator's classification of the generated image is represented by the discriminator's output (gan_output). The inputs (gen_noise and gen_label) and output (gan_output) of the GAN model are specified, and a new model (model) is created using the Model class.

The Adam optimizer is used to create the GAN model, which has a learning rate of 0.0002 and a beta value of 0.5. Binary cross-entropy, which is frequently utilized for GANs, is the loss function employed. It then returns the finished GAN model (model).

The GAN model was developed by connecting the generator and discriminator models. With the help of training the generator to create deceptive images, the GAN model aims to produce realistic images.



Get Label Input (in_label)
Embedding for Categorical Input (n_classes, 50)
Scale Up to Image Dimensions with Linear Activation (n_nodes)
Reshape to Additional Channel (in_shape[0], in_shape[1], 1)
Get Image Input (in_image)
Concatenate Label as a Channel (in_image, li)
Downsample (3×3, strides=(2,2), padding='same')
Downsample (3×3, strides=(2,2), padding='same')
Flatten Feature Maps ()
Dropout (0, 4)
Output (1, activation='sigmoid')
Define Model (in_image, in_label →out_layer)
Compile Model (loss='binary_crossentropy', optimizer=Adam(lr=0.0002, beta_1=0.5), metrics=['accuracy'])
Output

Figure 8: *A flowchart showing steps for a discriminator model architecture for cGAN model using tensor flow.*

The discriminator takes two inputs: an image input of shape in_shape (default is (32, 32, 3)) and a categorical label input of size 1. The discriminator's task is to classify whether the input image is real or generated by the generator.

The function define_discriminator takes two arguments: in_shape (shape of the input image) and n_classes (number of classes for the categorical label, default is 5). The label input (in_label) is defined as a 1-dimensional input. The categorical label is embedded into a continuous vector representation (li) using an embedding layer.

The embedding size is 50. The embedded label is then passed through a dense layer to match the dimensions of the input image. The output shape is (32, 32, 1) where 32x32 is the spatial dimension and 1 is the channel dimension. The image input (in_image) is defined with the specified shape (in_shape). The label tensor (li) and the image tensor (in_image) are concatenated along the channel dimension to create a tensor with 4 channels (3 channels for the image and 1 channel for the label). The concatenated tensor is passed through two convolutional layers with LeakyReLU activation and downsampling to learn hierarchical features.

The output shape after downsampling is (8, 8, 128). The feature maps are flattened and passed through a dropout layer for regularization. The flattened features are then passed through a dense layer with sigmoid activation to produce the discriminator's output, which is a probability indicating whether the input is real or generated. The model is instantiated with the inputs (in_image and in_label) and the output layer (out_layer), and then returned.

The discriminator model is compiled with the Adam optimizer using a learning rate of 0.0002 and a beta value of 0.5. The loss function used is binary cross-entropy, and the model is also evaluated using accuracy as a metric.

This architecture defines the discriminator model for the conditional GAN. It takes an image and a categorical label as inputs and outputs a probability indicating the authenticity of the input image. The model is trained to distinguish between real and generated images while considering the corresponding label information.

3.3.2 DCGAN using IDRID dataset

A generator model for a deep convolutional generative adversarial network (DC-GAN) is defined by this architecture. The generator attempts to produce realistic images using a random noise vector as input.

The tf.keras function make_generator_model builds a sequential model.Sequential(). The model begins with a dense layer that is entirely connected and has units of 7x7x256 (12544). A random noise vector is represented by the input shape (100,).

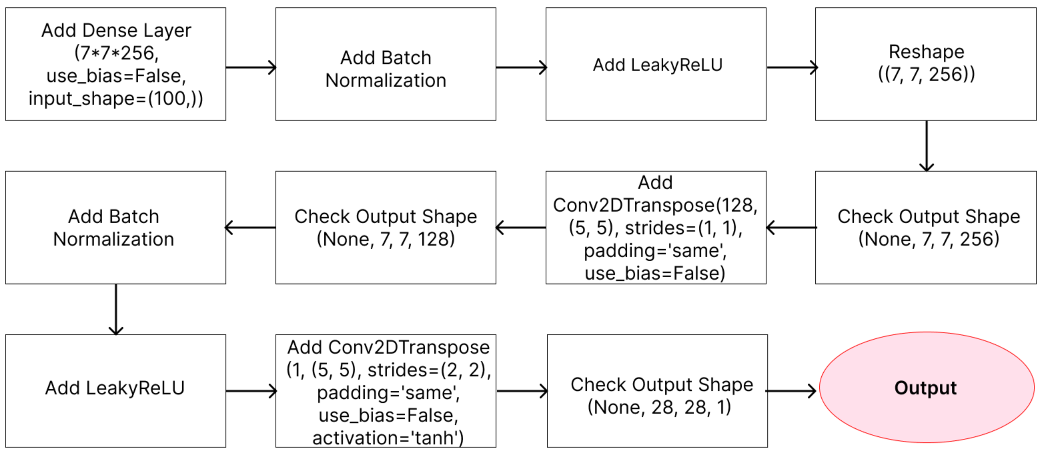

Add Dense Layer (7*7*256, use_bias=False, input_shape=(100,))
Add Batch Normalization
Add LeakyReLU
Reshape ((7, 7, 256))
Check Output Shape (None, 7, 7, 256)
Add Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_bias=False)
Check Output Shape (None, 7, 7, 128)
Add Batch Normalization
Add LeakyReLU
Add Conv2DTranspose (1, (5, 5), strides=(2, 2), padding='same', use_bias=False, activation='tanh')
Check Output Shape (None, 28, 28, 1)
Output

Figure 9: *The architectural flow of a DCGAN model generator using tensor flow.*

Use_bias is set to False in the argument. After the dense layer, batch normalization is used to normalize the activations and boost the model's performance and stability. Batch normalization is followed by the use of a LeakyReLU activation function. The model learns gradients more effectively and can handle vanishing gradients thanks to LeakyReLU.

The layers are used to transform the dense layer's output into a 7x7x256 tensor.function called Reshape. Convolutional layers with item A transposed (layers.With 128 filters, a kernel size of (5, 5), and a stride of (1, 1), Conv2DTranspose()) is added. Use_bias is set to False, and padding is set to'same'. Following the transposed convolutional layer, item Batch normalization and LeakyReLU activation are used to upsample the spatial dimensions to 7x7x128. With 64 filters, a kernel size of (5, 5), and a stride of (2, 2), another transposed convolutional layer is added. Use_bias is set to False, and padding is set to'same'. The spatial dimensions are further upsampled in this layer to 14x14x64. Afterwards, the second transposed convolutional layer is applied, followed by batch normalization and LeakyReLU activation.

The final step involves the addition of a transposed convolutional layer with 1 filter, kernel size of (5, 5), and stride of (2, 2). The activation function is set to 'tanh', use_bias is set to False, and padding is set to'same'. This layer generates the resulting image by upsampling the spatial dimensions to 28x28x1. To make sure the output shapes match the anticipated values, assertions are supplied.

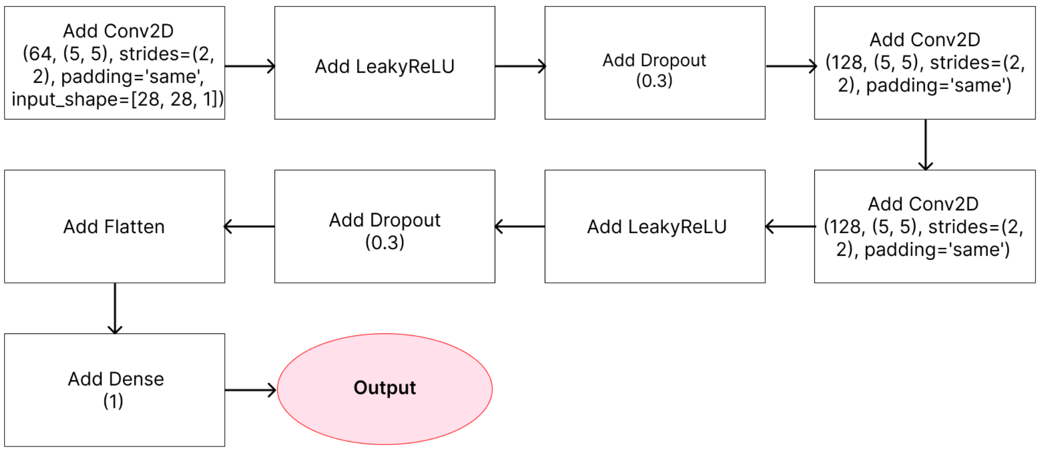

Add Conv2D (64, (5, 5), strides=(2, 2), padding='same', input_shape=[28, 28, 1])
Add LeakyReLU
Add Dropout (0.3)
Add Conv2D (128, (5, 5), strides=(2, 2), padding='same')
Add Conv2D (128, (5, 5), strides=(2, 2), padding='same')
Add LeakyReLU
Add Dropout (0.3)
Add Flatten
Add Dense (1)
Output

Figure 10: *The architectural flow of a DCGAN model discriminator using tensor flow.*

The next architecture defines a discriminator model for a deep convolutional generative adversarial network (DCGAN). The discriminator takes an input image and aims to classify whether it is real or generated by the generator.

The function make_discriminator_model creates a sequential model using tf.keras.Sequential(). The model starts with a convolutional layer (layers.Conv2D()) with 64 filters, a kernel size of (5, 5), and stride (2, 2). Padding is set to 'same', and the input shape is [28, 28, 1], representing a 28x28 grayscale image. This layer performs downsampling of the input image. A LeakyReLU activation function is applied after the convolutional layer. LeakyReLU helps the model to learn better gradients and handle vanishing gradients.

A dropout layer (layers.Dropout()) with a rate of 0.3 is added after the activation. Dropout helps to regularize the model and prevent overfitting. Another convolutional layer is added with 128 filters, a kernel size of (5, 5), and stride (2, 2). Padding is set to 'same'. This layer further downsamples the feature maps. LeakyReLU activation and dropout layers are applied after the second convolutional layer. The feature maps are flattened using layers.Flatten() to convert the 3D tensor into a 1D tensor.

A dense layer (layers.Dense()) with a single unit is added. This layer acts as the output layer of the discriminator, providing a single scalar output representing the probability of the input being real or generated.

This architecture defines a discriminator model with multiple convolutional layers that downsample the input image to learn hierarchical features. The model uses LeakyReLU activations and dropout layers for better gradient propagation and regularization. The final dense layer produces a single output that represents the discriminator's classification of the input as real or generated.

The PyTorch framework is used in the architecture to define a generator model. The generator seeks to produce artificial graphics from a random input vector.

The 'Generator' class is established, deriving from PyTorch's 'nn.Module' class. The generator and its layers are initialized using the constructor "__init," which is defined. The number of GPUs to use is represented by the input 'ngpu'. The main sequential module of the generator, which is made up of numerous convolutional transpose layers, is represented by the'self.main' variable.

'nn.ConvTranspose2d' is the initial layer, and it does a transpose convolution. It creates a feature map by upsampling the random input vector "Z." The output has a kernel size of 4, a stride of 1, and 'ngf * 8' channels. Padding is not present. The 'False' value for the 'bias' argument. After the initial transpose convolutional layer,

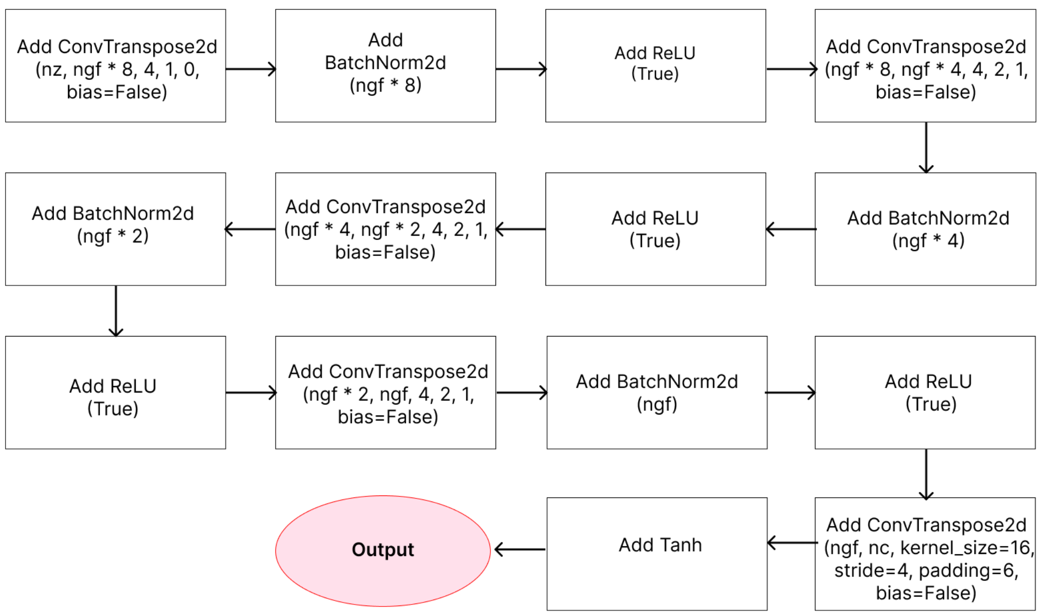

Add ConvTranspose2d (nz, ngf * 8, 4, 1, 0, bias=False)
Add BatchNorm2d (ngf * 8)
Add ReLU (True)
Add ConvTranspose2d (ngf * 8, ngf * 4, 4, 2, 1, bias=False)
Add BatchNorm2d (ngf * 4)
Add ReLU (True)
Add ConvTranspose2d (ngf * 4, ngf * 2, 4, 2, 1, bias=False)
Add BatchNorm2d (ngf * 2)
Add ReLU (True)
Add ConvTranspose2d (ngf * 2, ngf, 4, 2, 1, bias=False)
Add BatchNorm2d (ngf)
Add ReLU (True)
Add ConvTranspose2d (ngf, nc, kernel_size=16, stride=4, padding=6, bias=False)
Add Tanh
Output

Figure 11: *The architectural flow of a DCGAN model generator using pytorch.*

batch normalization ('nn.BatchNorm2d') is used to normalize the activations and increase model stability. After the batch normalization layer, a ReLU activation function ('nn.ReLU(True)') is used to add non-linearity.

The output from the first "nn.ConvTranspose2d" layer is added to the second layer, which further upsamples it. It has four channels with 'ngf * 4', a kernel size of 4, a stride of 2, and padding of 1. After the second transpose convolutional layer, batch normalization and ReLU activation are used. Two further transpose convolutional layers are added, gradually upsampling the feature maps, in a manner similar to the earlier phases. 'ngf' channels, a kernel size of 16, a stride of 4, and a padding of 6 are present in the final 'nn.ConvTranspose2d' layer. It seeks to produce the desired output image.

After the final convolutional layer, a Tanh activation function ('nn.Tanh()') is used to reduce the pixel values to the range [-1, 1], which represents the output image. The generator's forward pass is defined to be carried out via the 'forward' method. The main sequential module ('self.main') receives a vector of inputs called 'input' and processes it before returning the produced output.

In order to create a 64x64 image, a generator model is built using many transpose convolutional layers that gradually upsample the input random vector. The stability and quality of the generated images are enhanced using batch normalizing and ReLU activations. To verify that the pixel values are within the desired range, the Tanh activation function is applied to the output in the end.

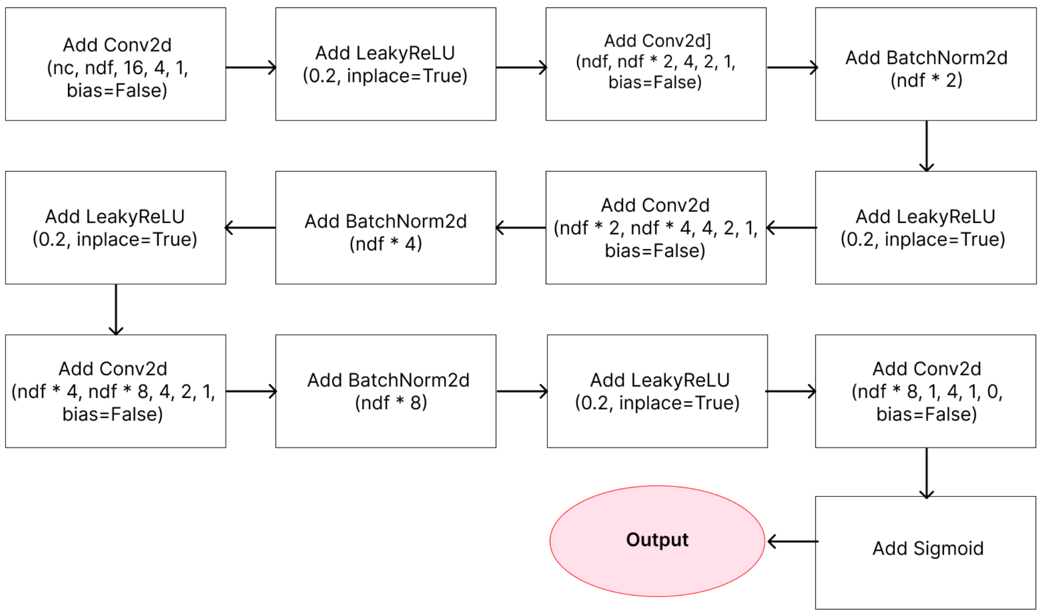

Add Conv2d (nc, ndf, 16, 4, 1, bias=False)
Add LeakyReLU (0.2, inplace=True)
Add Conv2d] (ndf, ndf * 2, 4, 2, 1, bias=False)
Add BatchNorm2d (ndf * 2)
Add LeakyReLU (0.2, inplace=True)
Add Conv2d (ndf * 2, ndf * 4, 4, 2, 1, bias=False)
Add BatchNorm2d (ndf * 4)
Add LeakyReLU (0.2, inplace=True)
Add Conv2d (ndf * 4, ndf * 8, 4, 2, 1, bias=False)
Add BatchNorm2d (ndf * 8)
Add LeakyReLU (0.2, inplace=True)
Add Conv2d (ndf * 8, 1, 4, 1, 0, bias=False)
Add Sigmoid
Output

Figure 12: *The architectural flow of a DCGAN model discriminator using pytorch.*

The following architecture defines a discriminator model using the PyTorch framework. The discriminator is responsible for distinguishing between real and fake images.

The ‘Discriminator‘ class is defined, inheriting from the ‘nn.Module‘ class provided by PyTorch. The constructor ‘__init__‘ is defined, which initializes the discriminator and its layers. It takes an argument ‘ngpu‘, which represents the number of GPUs to use. The ‘self.main‘ variable represents the main sequential module of the discriminator, which consists of a series of convolutional layers.

The first layer is a ‘nn.Conv2d‘ layer, which performs a convolution. It takes the input image with ‘nc‘ channels, a kernel size of 16, stride of 4, and padding of 1. The ‘bias‘ parameter is set to ‘False‘. A LeakyReLU activation function (‘nn.LeakyReLU(0.2, inplace=True)‘) is applied after the first convolutional layer to introduce non-linearity. The ‘0.2‘ specifies the negative slope for the negative input region. The second ‘nn.Conv2d‘ layer is added, which takes the output from the previous layer and performs another convolution. It has ‘ndf‘ channels, a kernel size of 4, stride of 2, and padding of 1.

Batch normalization (‘nn.BatchNorm2d‘) is applied after the second convolutional layer to normalize the activations and improve the stability of the model. LeakyReLU activation is applied after the batch normalization layer. Similar to the previous steps, two more convolutional layers are added, gradually reducing the spatial dimensions of the feature maps while increasing the number of channels. The last ‘nn.Conv2d‘ layer has ‘ndf * 8‘ channels, a kernel size of 4, stride of 1, and no padding. It aims to produce a single-channel output representing the probability of the input image being real or fake. A Sigmoid activation function (‘nn.Sigmoid()‘) is applied after the last convolutional layer to squash the output values to the range [0, 1], representing the probability. The ‘forward‘ method is defined to perform the forward pass of the discriminator. It takes an input image ‘input‘ and passes it through the main sequential module (‘self.main‘), returning the discriminator’s prediction.

A discriminator model is generated with several convolutional layers that progressively downsample the input image and output a single probability value. LeakyReLU activations and batch normalization are used to improve the discriminative capacity and stability of the discriminator. The Sigmoid activation function is applied to the final output to ensure the output is within the range [0, 1], representing the probability of the input image being real.

4 Result & Analysis

4.1 Dataset

We used the Indian Diabetic Retinopathy Image Dataset (IDRID) in our thesis experimentation. The INDIAN DIABETIC RETINOPATHY IMAGE DATASET (IDRID) [5] is a collection of medical images of the retina that was created to support the development of machine learning algorithms for the diagnosis of diabetic retinopathy. The IDRID dataset contains images of the retina from individuals with and without diabetic retinopathy, as well as images of various stages of the disease.

The IDRID dataset was created by a team of researchers in India as part of a larger effort to improve the diagnosis and management of diabetic retinopathy. The dataset is intended to be a resource for the development and evaluation of machine learning algorithms for the diagnosis of diabetic retinopathy, and it is widely used in the field of medical image analysis.

The IDRID Dataset is available for download by researchers and clinicians who are interested in using it for machine learning or other research purposes.

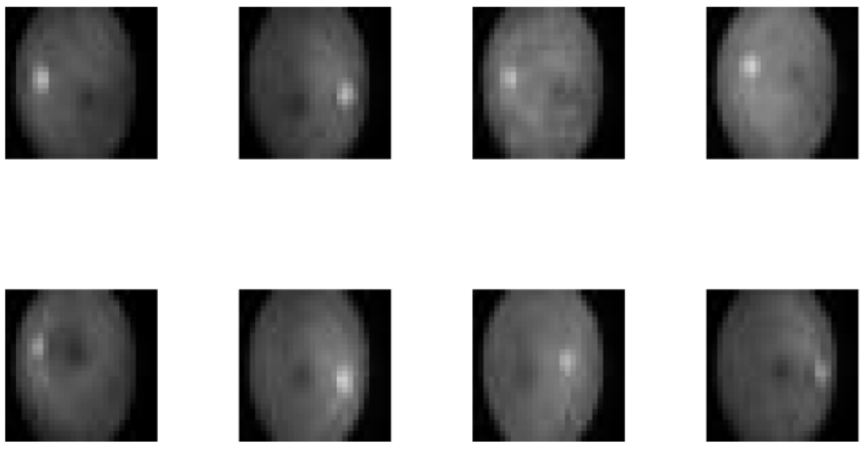


Figure 13: Some data samples from IDRID Dataset

4.2 Evaluation Metrics

For the model evaluation, Peak signal to noise ratio (PSNR) and structural index similarity (SSIM) values are calculated through comparison of generated images using the trained model with the original images from the dataset.

Peak Signal to Noise Ratio (PSNR) [10]: The term peak signal-to-noise ratio (PSNR) is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. The formula for calculating PSNR value is as follows:

$$\text{Mean Squared Error, MSE} = \frac{1}{MN}\sum_{n=0}^{M}\sum_{m=1}^{N}[g(n,\hat{m}) - g(n,m)]^2$$

$$PSNR = 10\log_{10}\left(\frac{\text{peakval}^2}{\text{MSE}}\right)$$

Structural Similarity Index Measure (SSIM) [10]: The structural similarity index measure (SSIM) is a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos. SSIM is used for measuring the similarity between two images.

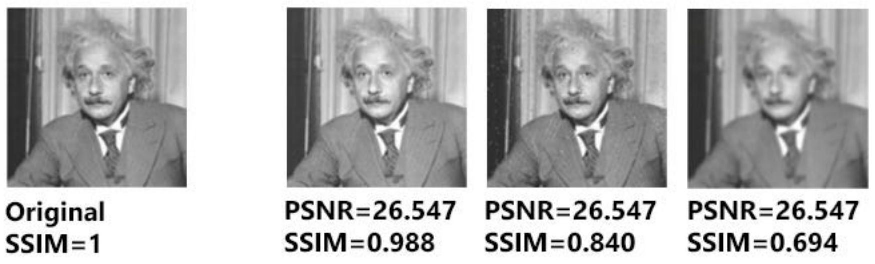


Figure 14: *An example of PSNR and SSIM calculation of similar looking images.*

Structural Similarity Index Method can be expressed through these three terms as:

SSIM(x,y) $= [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$
Here, l is the luminance (used to compare the brightness between two images), c is the contrast (used to differ the ranges between the brightest and darkest region of two images) and s is the structure (used to compare the local luminance pattern between two images to find the similarity and dissimilarity of the images) and α, β and γ are the positive constants [11].

Again luminance, contrast and structure of an image can be expressed separately as:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1},$$
$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2},$$
$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3},$$

Where μ_x and μ_y are the local means, σ_x and σ_y are the standard deviations and $\sigma_x y$ is the cross-covariance for images x and y sequentially. If $\alpha = \beta = \gamma = 1$, then the index is simplified as the following form :

$$SSIM(x,y) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right) \cdot \left(\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\right) \cdot \left(\frac{\mu_x^2 + \mu_y^2 + C_1}{\sigma_x^2 + \sigma_y^2 + C_2}\right)$$

4.3 Hyperparameters

In generative models, such as Generative Adversarial Networks (GANs), the hyperbolic tangent (tanh) activation function is frequently chosen as the generator's activation function. For some jobs involving data normalization or creation, the tanh activation function transfers the generator's output to a range between -1 and 1.

Regarding the loss function, GANs frequently employ binary cross-entropy for binary classification tasks like differentiating between genuine and generated data. The dissimilarity between the anticipated output and the goal label (0 or 1) for each sample is measured by binary cross-entropy loss.

In the domain of medical image synthesis, the discriminator in a GAN seeks to distinguish between genuine and created images while the generator in the GAN strives to produce realistic medical images. The binary cross-entropy loss function encourages the generator to produce images that the discriminator is more likely to categorize as real, which can aid in the training process.

It's crucial to remember that the selection of the activation function and loss function may change based on the particular specifications and features of the task involving the synthesis of medical images. Depending on the intended results and the GAN model's structure, additional activation functions (like ReLU or Leaky ReLU) and loss functions (like Wasserstein loss or feature matching loss) may also be utilized.

4.4 Results

4.4.1 Quantitative Result

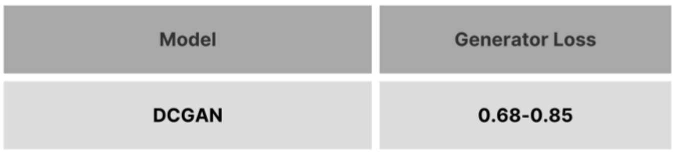
After running the DCGAN model using tensorflow with the pre-processed data set at hand - the IDRID dataset - we get the follwing results:

The training loop iterates over the IDRID dataset in batches, with each batch containing 128 samples. After each iteration, the code prints the sizes of the tensors used in the training process. These tensor sizes provide insights into the dimensions of the data being processed. The training progress is displayed in the format [current_epoch/total_epochs][current_batch/total_batches]. This indicates the progress of the training process in terms of epochs and batches. The [0/5][0/2] represents the current progress of the training in terms of epochs and batches. In this example, it shows that the current epoch is 0 out of 5 total epochs, and the current batch is 0 out of 2 total batches.

The discriminator loss (‘Loss_D‘) and generator loss (‘Loss_G‘) are printed. These values represent the calculated losses for the discriminator and generator networks, respectively. The discriminator loss indicates how well the discriminator can distinguish between real and fake data, while the generator loss indicates how well the generator can fool the discriminator.

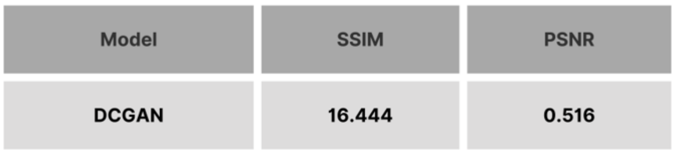
The average discriminator output for real data (‘D(x)‘) is displayed. This value represents the average prediction made by the discriminator for real data samples in the current batch. It shows the discriminator’s confidence in classifying real data. D(x): 0.2162 value represents the average output of the discriminator (D) for real data samples (x) in the current batch. The average discriminator output for generated data (‘D(G(z))‘) is also printed. This value represents the average prediction made by the discriminator for generated (fake) data samples in the current batch. It indicates the discriminator’s confidence in classifying generated data.

D(G(z)): 0.4134 / 0.1291: These values represent the average output of the discriminator (D) for generated data samples (G(z)) and real data samples (x) in the current batch. The first value (0.4134) indicates how well the discriminator is classifying generated data, while the second value (0.1291) represents its classification of real data. Loss_D: 2.2727 Loss_G: 2.3577 values represent the discriminator loss (Loss_D) and generator loss (Loss_G) at the initial steps which gradually decreases for the discriminator loss and increases for the generator loss to Loss_D: 0.1340 Loss_G: 6.8828. The tensor sizes are printed again to provide a comparison before and after the calculations. This allows for tracking any changes in tensor dimensions during the training process.



Model	Generator Loss
DCGAN	0.68-0.85

Figure 15: *The loss calculation from the trained model.*



Model	SSIM	PSNR
DCGAN	16.444	0.516

Figure 16: *The PSNR and SSIM calculation from the trained model.*

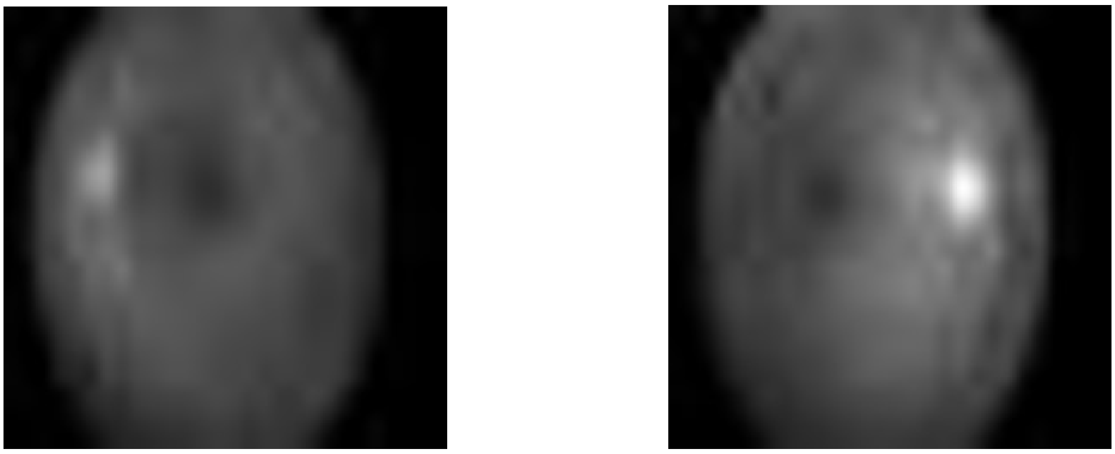

(a) Real Image (b) Fake Image

Figure 17: *The difference between generated(fake) and original(real) image after medical image synthesis using DCGAN model.*

4.4.2 Qualitative Result

The generated images have been compared to the entire batch of the original dataset i.e., IDRID dataset and the comparison result has been evaluated through SSIM and PSNR values. The SSIM and PSNR value obtained for a randomly picked image against the original dataset was 0.516 and 16.444 respectively. Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, where higher is better. Values over 40 dB are normally considered very good and those below 20 dB are normally unacceptable [12]. An SSIM score of 1.00 indicates perfect structural similarity, as is expected out of identical images [12].

The progress of GAN training by showing the tensor sizes, loss values, and discriminator predictions for real and generated data is the main data displayed as result for the implemented DCGAN model using pytorch. Monitoring these metrics helps assess the performance and convergence of the discriminator and generator networks throughout the training process.

5 Conclusion

GANs are deep learning models consisting of a generator and a discriminator. They are employed in various domains, including medical image processing. GANs can generate synthetic medical images resembling real data, which aids in augmenting limited training data for other machine learning models. This improves the performance of tasks like diagnosis and prognosis. GANs also enhance medical image quality by removing noise or artifacts and enhancing resolution.

Our objective was to develop a robust conditional GAN that can generate realistic medical image data. This approach aims to address the imbalance between malignant and benign cases, thereby improving the accuracy of disease prediction models. To achieve this, we utilized the IDRID dataset. By leveraging this dataset, we generated synthetic images that closely resemble real medical images, effectively augmenting the existing data. This augmentation will increase the dataset size, resulting in a more balanced representation of malignant and benign cases. The expanded dataset will contribute to training more accurate disease prediction models.

The DCGAN model was trained on the IDRID dataset using PyTorch. The training loop processed the dataset in batches of 128 samples, displaying tensor sizes and tracking the progress in terms of epochs and batches. The discriminator and generator losses were printed, indicating the performance of the networks. Average discriminator outputs for real and generated data were shown, along with their respective classifications. The model evaluation involved calculating PSNR and SSIM values to assess image quality. Overall, the DCGAN model trained on the IDRID dataset yielded results in terms of loss values, discriminator outputs, and evaluation metrics like PSNR and SSIM.

A future goal for our research on medical image synthesis using GAN architectures is to develop more robust and novel GAN architectures and training techniques that can generate high-fidelity and clinically relevant medical images across multiple modalities. Additionally, focusing on addressing challenges such as data scarcity, class imbalance, and interpretability of synthesized images would be important for advancing the field. Ultimately, the goal is to create GAN models that can generate synthetic medical images with such accuracy and realism that they can be seamlessly integrated into clinical practice, aiding in diagnosis, treatment planning, and medical research.

References

[1] Youssef Skandarani, Pierre-Marc Jodoin, and Alain Lalande. Gans for medical image synthesis: An empirical study. *Journal of Imaging*, 9(3):69, 2023.

[2] Meiqin Gong, Siyu Chen, Qingyuan Chen, Yuanqi Zeng, and Yongqing Zhang. Generative adversarial networks in medical image processing. *Current Pharmaceutical Design*, 27(15):1856–1868, 2021.

[3] Tonghe Wang, Yang Lei, Yabo Fu, Jacob F Wynne, Walter J Curran, Tian Liu, and Xiaofeng Yang. A review on medical imaging synthesis using deep learning and its clinical applications. *Journal of applied clinical medical physics*, 22(1):11–36, 2021.

[4] Saisai Ding, Jian Zheng, Zhaobang Liu, Yanyan Zheng, Yanmei Chen, Xiaomin Xu, Jia Lu, and Jing Xie. High-resolution dermoscopy image synthesis with conditional generative adversarial networks. *Biomedical Signal Processing and Control*, 64:102224, 2021.

[5] Siddharth Gupta, Avnish Panwar, Silky Goel, Ankush Mittal, Rahul Nijhawan, and Amit Kumar Singh. Classification of lesions in retinal fundus images for diabetic retinopathy using transfer learning. In *2019 international conference on information technology (ICIT)*, pages 342–347. IEEE, 2019.

[6] Sandeep B Somvanshi and Nanasaheb D Thorat. Introduction to imaging modalities. In *Advances in Image-Guided Cancer Nanomedicine*. IOP Publishing, 2022.

[7] Hamed Alqahtani, Manolya Kavakli-Thorne, and Gulshan Kumar. Applications of generative adversarial networks (gans): An updated review. *Archives of Computational Methods in Engineering*, 28:525–552, 2021.

[8] Y Sravani Devi and S Phani Kumar. Dr-dcgan: A deep convolutional generative adversarial network (dc-gan) for diabetic retinopathy image synthesis. *Webology (ISSN: 1735-188X)*, 19(2), 2022.

[9] Sandra Carrasco Limeros, Sylwia Majchrowska, Mohamad Khir Zoubi, Anna Rosén, Juulia Suvilehto, Lisa Sjöblom, and Magnus Kjellberg. Gan-based generative modelling for dermatological applications–comparative study. *arXiv preprint arXiv:2208.11702*, 2022.

[10] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 7(3):8–18, 2019.

[11] Rohit Kumar and Vishal Moyal. Visual image quality assessment technique using fsim. *International Journal of Computer Applications Technology and Research*, 2(3):250–254, 2013.

[12] David R Bull. Digital picture formats and representations. *Communicating pictures*, pages 99–132, 2014.