Implementation and Evaluation of Diffusion model with U-Net versus traditional DC-GANs as Synthetic Image Generators

Param Chordiya
Electrical & Computer Engineering
PID # A69026988

Riya Joshi
Electrical & Computer Engineering
PID # A59023015

Abstract

Generative Adversarial Networks (GANs) have revolutionized synthetic image generation. This project evaluates the performance of two generative models, Diffusion Models with U-Net architecture and Deep Convolutional GANs (DCGANs), for COVID-19 lung image synthesis. The models are implemented and optimized through hyperparameter tuning. The Diffusion Model denoises samples from Gaussian noise to produce realistic images, while the DCGAN uses adversarial learning to generate synthetic images. Performance is tracked by measuring training times and plotting losses per epoch.

Image quality is assessed using the Frechet Inception Distance (FID) score, and visual comparisons are made. This comparative analysis provides insights into the strengths and weaknesses of each approach.

1 Introduction

1.1 Motivation

The COVID-19 epidemic has underlined the crucial need for precise and timely diagnosis methods. Medical imaging, including lung imaging with CT scans and X-rays, has been critical in identifying and monitoring disease progression. However, the scarcity of high-quality annotated medical pictures creates a huge difficulty for developing robust machine learning models. Synthetic picture synthesis is a potential technique that augments current datasets, enhancing model performance and generalization.

1.2 Problem

The key issue we want to address is the creation of high-quality synthetic lung images from COVID-19 patients. These synthetic images can enrich real-world datasets, assisting in the creation and testing of diagnostic models. Traditional data augmentation strategies, while useful, frequently fall short of producing realistic and diverse samples. Generative models, like Generative Adversarial Networks (GANs) and Diffusion Models, have demonstrated amazing success in producing realistic images in a variety of fields. This study studies how advanced generative models can be applied to the specific difficulty of synthetic medical picture production as well as understand which one of them performs better.

1.3 Approach

Our approach entails developing and comparing two cutting-edge generative models: a U-Net-based Diffusion Model and a Deep Convolutional GAN (DCGAN). The Diffusion Model employs

a U-Net architecture to gradually denoise samples from Gaussian noise, yielding realistic image production. In contrast, the DCGAN is made up of a generator and a discriminator who are engaged in an adversarial learning process, with the generator aiming to produce realistic images and the discriminator evaluating their authenticity. Key components of our approach include:

- **Dataset**: Using a COVID-19 lung image dataset for training both models.
- **Model Implementation**: Detailed construction of the U-Net based Diffusion Model and DCGAN.
- **Hyperparameter Tuning**: Extensive tuning of learning rates and optimization parameters to enhance model performance.
- **Performance Tracking**: Monitoring training times and losses for both models to assess their efficiency and stability.

2 Related Work

The field of synthetic image generation has experienced significant advancements, particularly with the development of Generative Adversarial Networks (GANs) and Diffusion Models.

Goodfellow et al. introduced GANs in 2014, and they have since become a cornerstone of generative modeling. GANs are made up of two neural networks, a generator and a discriminator, that are trained adversarially to produce and assess synthetic data. [1]. Radford et al. (2015) presented the Deep Convolutional GAN (DCGAN), which includes architectural advances such as the use of convolutional layers without pooling, batch normalization, and ReLU activations, considerably enhancing the stability and quality of generated images. [2]. Because of their ability to produce high-resolution images, DCGANs have become the industry standard in generative tasks. GANs have been widely employed in medical imaging. Nie et al. (2017) used GANs to synthesis CT images from MRI scans, facilitating multimodal medical image analysis [3]. Frid-Adar et al. (2018) used GANs to augment limited datasets of liver lesions, which improved the performance of a classifier trained on synthetic data [4].

Ho et al. (2020) defined the diffusion process in their Denoising Diffusion Probabilistic Models (DDPMs), which outperformed other picture synthesis challenges. The usage of U-Net architecture with skip connections was critical in maintaining high-resolution details during the denoising processes [5]. Diffusion models have showed potential in medical imaging, particularly for image denoising and synthesis. Song et al. (2020) expanded the usage of DDPMs to medical image production, obtaining cutting-edge findings in generating high-fidelity images of human brains [6].

Karras et al. (2019) created StyleGAN, an advanced GAN architecture that produced exceptional image quality and diversity. Comparative tests employing criteria such as the Frechet Inception Distance (FID) score revealed that StyleGAN outperformed typical GANs in generating photorealistic images [7]. Dhariwal and Nichol (2021) conducted comprehensive comparisons between GANs and diffusion models, concluding that diffusion models typically yield lower FID values, indicating improved picture quality [8].

Training stability is a major concern with GANs. Arjovsky et al. (2017) proposed the Wasserstein GAN (WGAN) to increase training stability by employing the Wasserstein distance as a loss function [9]. Diffusion Models, as established by Nichol and Dhariwal (2021), provide more stable training but demand significantly more computer resources [10]

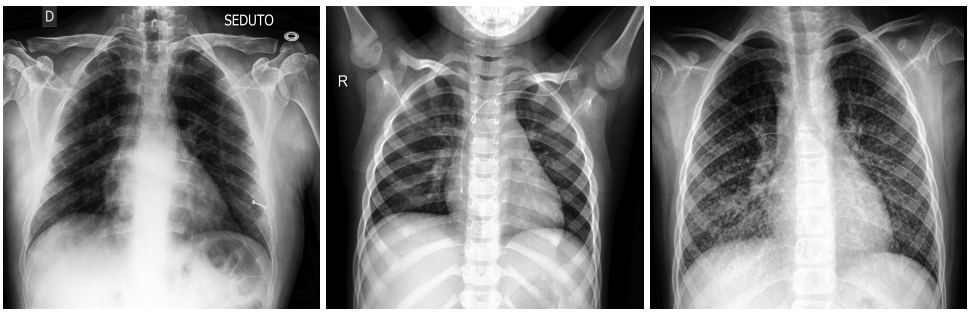
3 Method

3.1 Data Prepration

The dataset used in this study was obtained from Kaggle. This dataset is freely available, in accordance with open science principles, and it promotes reproducibility and transparency. The dataset, named Covid19-dataset, has been curated to address a variety of Covid-19-related topics. For the purpose of this study, only the training portion of the Covid-19 dataset was used. This subset is used to train the Diffusion Model as well as the Generative Adversarial Networks (GANs) to create synthetic images similar to those seen in the dataset. The Covid-19 Chest X-rays collection includes images divided

into three categories: Covid, Normal, and Viral Pneumonia. Each class reflects a distinct medical problem connected to chest X-ray imaging.

The dataset contains 111 photos classified as Covid, 70 images classed as Normal, and 70 images labeled as Viral Pneumonia for training. The test set includes 26 photos for Covid and 20 images for Normal and Pneumonia. Images go through preprocessing stages to ensure they are suitable for model training. These stages involve resizing to 64x64 pixels, center cropping for consistency. Each image was normalized to have pixel values between 0 and 1, improving the stability of the training process. The following are some instances of images from the dataset, as seen in Figure. 1



(a) Covid Affected Lungs (b) Normal Lungs (c) Pneumonia Affected Lungs

Figure 1: Different condition Lungs as seen in chest X-rays [11]

3.2 The U-Net Architecture

The U-Net architecture is a convolutional network intended for biomedical image segmentation, distinguished by its encoder-decoder structure and skip connections. Because of its symmetrical design, this building is especially excellent at producing high-resolution photographs. The encoder compresses the input image into a lower-dimensional feature representation by applying a sequence of convolutional layers followed by max-pooling processes. The decoder reconstructs the image from encoded information using transposed convolutions, eventually restoring the original resolution. These links between the encoder and decoder layers enable the model to preserve high-resolution features, resulting in higher-quality output images. The following pseudocode Algorithm 1 illustrates the U-Net architecture used in the diffusion model:

Algorithm 1 U-Net Architecture for Diffusion Model

Input: Noisy image x_t
Output: Denoised image x_0
function U-NET(x_t)
    $h_1 \leftarrow \text{ConvBlock}(x_t)$
    $h_2 \leftarrow \text{ConvBlock}(h_1)$
    $h_3 \leftarrow \text{ConvBlock}(h_2)$
    $h_4 \leftarrow \text{ConvBlock}(h_3)$
    $b \leftarrow \text{Bottleneck}(h_4)$
    $u_4 \leftarrow \text{UpConvBlock}(b) + h_4$
    $u_3 \leftarrow \text{UpConvBlock}(u_4) + h_3$
    $u_2 \leftarrow \text{UpConvBlock}(u_3) + h_2$
    $u_1 \leftarrow \text{UpConvBlock}(u_2) + h_1$
    $x_0 \leftarrow \text{Conv}(u_1)$
    return x_0
end function

U-Net as proposed by Ronneberger et al. for biomedical image segmentation can be seen in the Figure 2

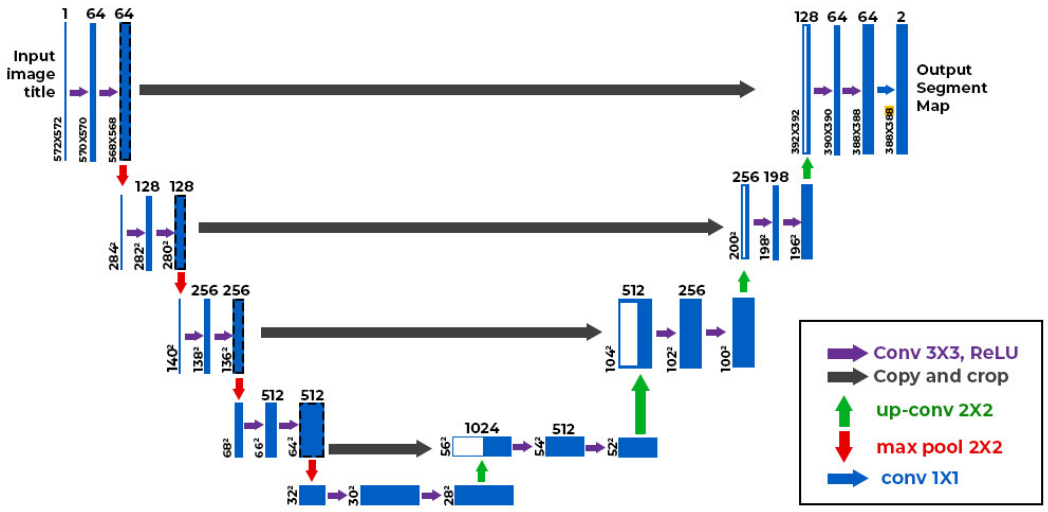


Figure 2: U-net architecture as shown by Ronneberger et. al. [12]

3.3 Weights Initialization

The stability and efficiency of training deep neural networks rely heavily on proper network weight initialization. Weights for both the U-Net-based Diffusion Model and the DCGAN were initialized using the Xavier (Glorot) initialization approach, which ensures that the weights are scaled appropriately to the size of the network layers, resulting in smoother gradients during backpropagation. The pseudocode 2 below represents the implementation of weights initialization

Algorithm 2 Xavier Initialization

function XAVIERINIT(W)
  $d \leftarrow \sqrt{\frac{6}{\text{fan\_in}+\text{fan\_out}}}$
  $W \leftarrow \text{Uniform}(-d, d)$
  return W
end function

3.4 Model Training

The training process for both the diffusion model as well as the DC-GANs followed a similar structure to maintain consistency.

3.4.1 Training Diffusion Model

The diffusion model learns to reconstruct clean images from noisy input while making the model learn from progressively demonising the images. Starting with initialization of Xavier weights as the U-Net Diffusion parameters, which also helps in reaching a stable convergence as weights get scaled in accordance with the layer dimensions. During each iteration of training, Gaussian Noise is added to the input images to create samples that are noisy. Followed by this, we want our U-Net model to denoise the noisy images iteratively. As for the loss function, we make use of something called as perceptual loss added to our Mean Squared Error Loss (MSE) between the ground truth images and the generated images. We try to minimize this loss. In short we can imagine the Algorithm of training as follows shown in Algorithm 3 below.

Algorithm 3 Training Algorithm for Diffusion Model

for each epoch **do**
  for each batch **do**
    $x_0 \leftarrow$ Real Image Batch
    $x_t \leftarrow$ Add Gaussian Noise to x_0
    $x_{\text{pred}} \leftarrow$ U-Net(x_t)
    $loss \leftarrow$ MSE(x_{pred}, x_0)
    Backpropagate and update parameters
  end for
end for

3.4.2 Training DC-GANs

The training for DC-GANs involved two parts as adversarial process. We mainly train the generator and the discriminator alternatively. The generator is aimed at producing images close to real images from random noise and the discriminator has to effectively categorize these images into real or fake. Both networks were initialized using the Xavier method to ensure stable training. The generator training method starts by creating a set of synthetic images from random noise inputs. These images were passed into the discriminator, which returned real or fake for the input images. The generator loss, determined using binary cross-entropy, indicated how well the generator fooled the discriminator. To minimize this loss, the generator settings were changed with the Adam optimizer.

The discriminator is trained using both real and synthetic images. The training dataset provided real images, whereas the updated generator produced synthetic images. The discriminator examined both sets of photos and made predictions about their legitimacy. The discriminator loss, which was similarly calculated using binary cross-entropy, accounted for its ability to correctly identify real and fraudulent images. The discriminator parameters are changed to minimize the loss and improve the capacity to discriminate between actual and generated images. This adversarial training procedure continues for a set number of epochs, with the generator and discriminator trained alternately. Throughout the training process, parameters such as generator and discriminator losses were monitored to ensure the models' progress and stability. The trained models were then tested on a distinct set of lung images, with image quality measured using the Frechet Inception Distance (FID) score and visual inspections to provide a full evaluation of their performance.

Algorithm 4 below demonstrated the applied training process in our work.

Algorithm 4 Training Algorithm for DCGAN

for each epoch **do**
  for each batch **do**
    $z \leftarrow$ Random Noise Batch
    $x_{\text{fake}} \leftarrow$ Generator(z)
    $y_{\text{fake}} \leftarrow$ Discriminator(x_{fake})
    $g_{\text{loss}} \leftarrow$ Binary Cross-Entropy$(y_{\text{fake}}, \text{Real Labels})$
    Backpropagate and update generator parameters
    $x_{\text{real}} \leftarrow$ Real Image Batch
    $y_{\text{real}} \leftarrow$ Discriminator(x_{real})
    $y_{\text{fake}} \leftarrow$ Discriminator$(x_{\text{fake}}.detach())$
    $d_{\text{loss}} \leftarrow$ Binary Cross-Entropy$(y_{\text{real}}, \text{Real Labels})$ + Binary Cross-Entropy$(y_{\text{fake}}, \text{Fake Labels})$
    Backpropagate and update discriminator parameters
  end for
end for

3.4.3 Model Evaluation

For the evaluation of model, we use both qualitative and quantitative evaluation. The Frechet Inception Distance also known as FID score was used as the primary metric for evaluation of the quality of

the generated images. The FID Score calculates the distance between the feature representation of generated and real images.

Algorithm 5 FID Score Calculation

function CALCULATEFID($\mu_r, \Sigma_r, \mu_g, \Sigma_g$)
  FID $\leftarrow ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$
  return FID
end function

4 Experiments

We perform the experimentation first on Diffusion model followed by experimentation on a DC-GANs model. We also update our U-NET in Diffusion model to be more complex than a vanilla diffusion model as with DC-GANs we modify our Adversarial network to be deeper for better results.

4.1 About Dataset

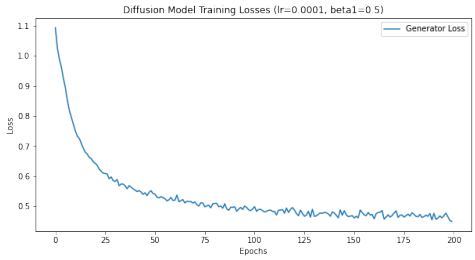
The dataset used in this study was obtained from Kaggle. This dataset is freely available, in accordance with open science principles, and it promotes reproducibility and transparency. The dataset, named Covid19-dataset, has been curated to address a variety of Covid-19-related topics [11]

The dataset contains 111 photos classified as Covid, 70 images classed as Normal, and 70 images labeled as Viral Pneumonia for training. The test set includes 26 photos for Covid and 20 images for Normal and Pneumonia. Images go through preprocessing stages to ensure they are suitable for model training. These stages involve resizing to 64x64 pixels, center cropping for consistency. Each image was normalized to have pixel values between 0 and 1, improving the stability of the training process.

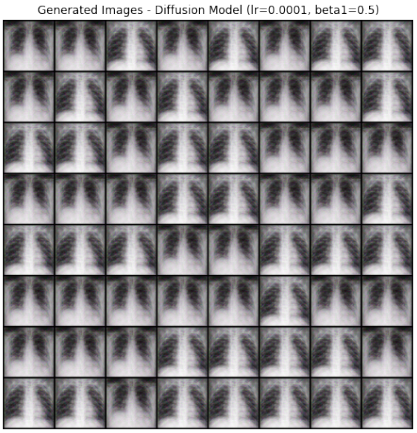
4.2 U-Net Diffusion Model Implementation

We first implement a basic diffusion model and modify it to U-Net based diffusion model. Followed by which we perform hyper-parameter tuning on the model. We select the best model based on the FID Score while we train our model based on reducing strategy of a modified loss type which is a combination of Mean Squared Error and Perceptual Loss. Some of the experimentation loss plots and respective image generation examples are as follows.

We aimed at tuning the learning rate and beta1 values as hyper-parameters. The results of combination of learning rate 0.0001 and beta1 0.5 are observed in Figure 3, while on increasing beta1 too 0.7 we the results are in Figure 4 . Similarly, for learning rate 0.001 and beta1 0.5, Figure 5 represents the output and with same learning rate but beta1 as 0.7 we observe the loss and generation in Figure 6.

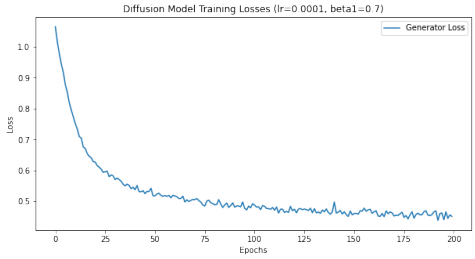


(a) Generator Loss

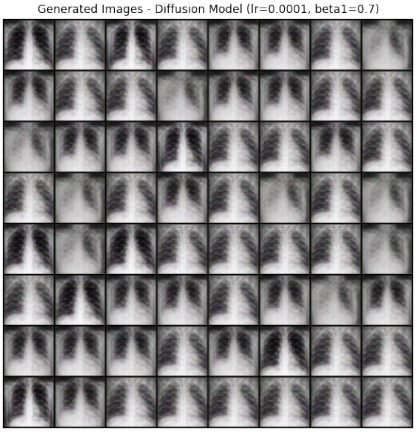


(b) Generated Images

Figure 3: Learning Rate: 0.0001 & beta1: 0.5

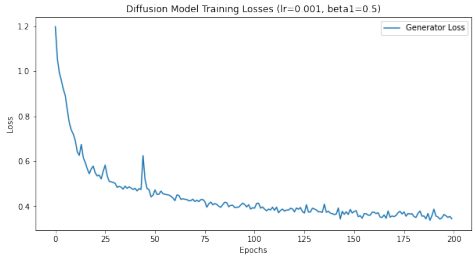


(a) Generator Loss

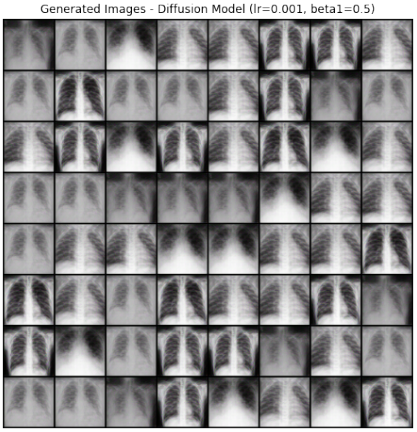


(b) Generated Images

Figure 4: Learning Rate: 0.0001 & beta1: 0.7

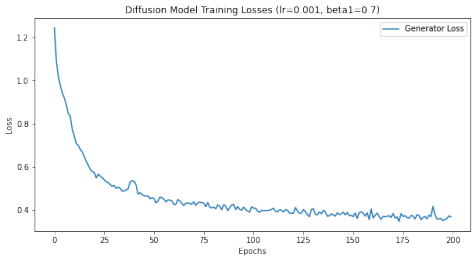


(a) Generator Loss

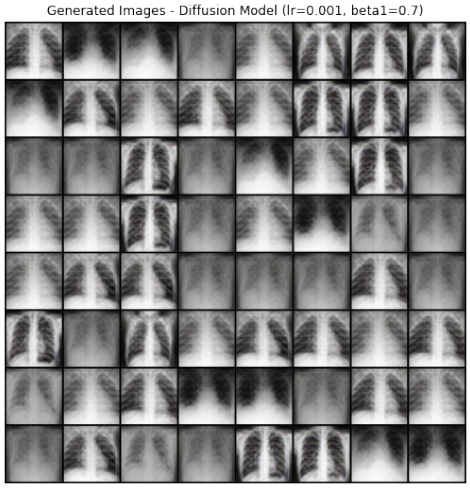


(b) Generated Images

Figure 5: Learning Rate: 0.001 & beta1: 0.5



(a) Generator Loss



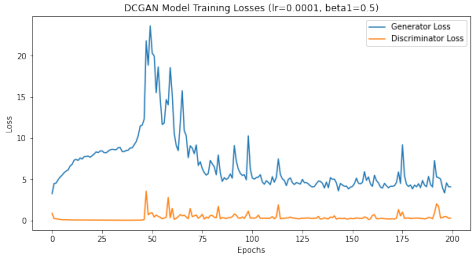
(b) Generated Images

Figure 6: Learning Rate: 0.001 & beta1: 0.7

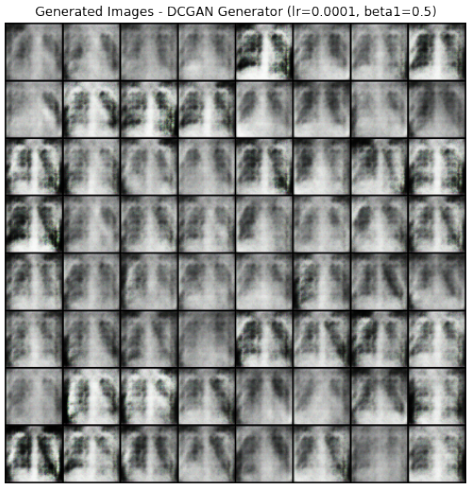
4.3 DC-GANs Model Implementation

In this section, we implemented our DC-GANs model and evaluated it on FID Score. We performed hyper-parameter tuning on this as well and considered the same hyper-parameters as diffusion model to maintain consistency and comparability in the results.

The results of combination of learning rate 0.0001 and beta1 0.5 are observed in Figure 7, while on increasing beta1 too 0.7 we the results are in Figure 8 . Similarly, for learning rate 0.001 and beta1 0.5, Figure 9 represents the output and with same learning rate but beta1 as 0.7 we observe the loss and generation in Figure 10.

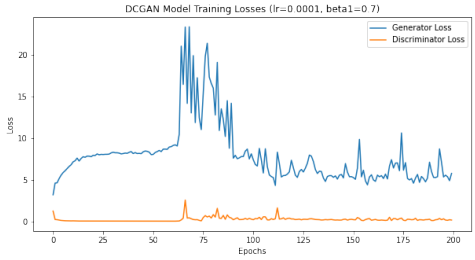


(a) Generator Loss

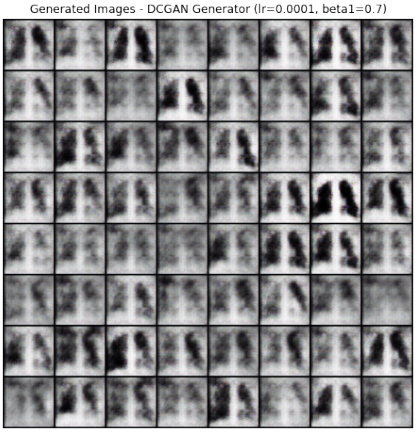


(b) Generated Images

Figure 7: Learning Rate: 0.0001 & beta1: 0.5

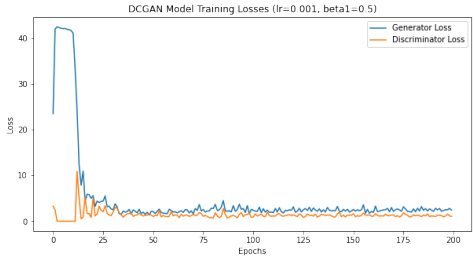


(a) Generator Loss

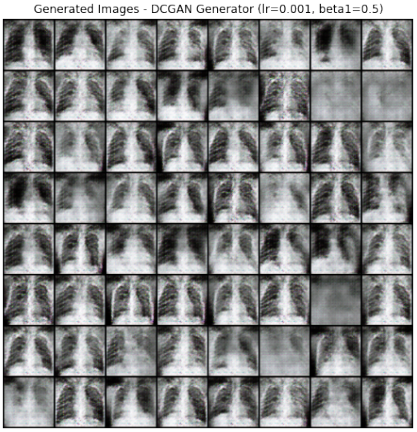


(b) Generated Images

Figure 8: Learning Rate: 0.0001 & beta1: 0.5

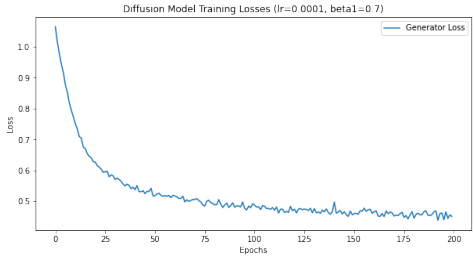


(a) Generator Loss

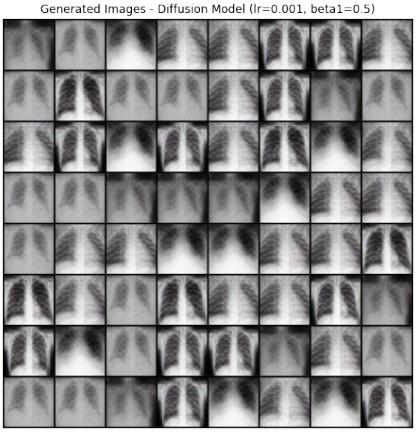


(b) Generated Images

Figure 9: Learning Rate: 0.0001 & beta1: 0.5



(a) Generator Loss



(b) Generated Images

Figure 10: $LearningRate$: 0.0001 & $beta1$: 0.5

5 Results

Based on our evaluation and observation we curate the final results based on the FID scores as well as Inception Scores. Since our model selection is based on FID scores, the diffusion model with learning rate of 0.001 and a beta1 value of 0.7. While in case of DC-GANs, we get the best model on learning rate of 0.001 and beta1 of 0.7 as well.

On further training the models on the best parameters for 250 epochs each, we get the loss plots as in Figure 11 and Figure for Diffusion model and DC-GANs respectively.

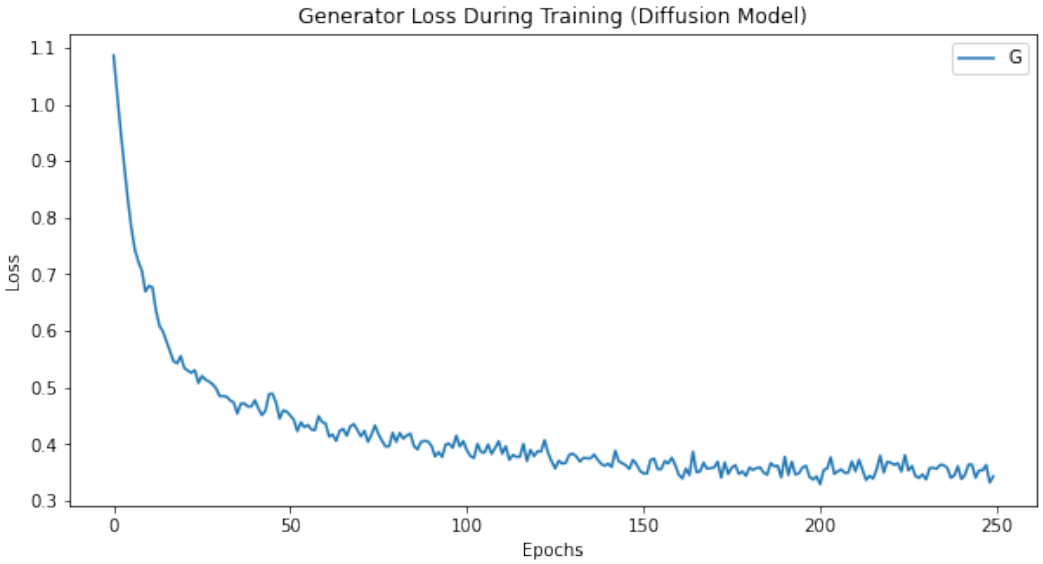


Figure 11: Diffusion Model Loss Plot for best Hyper-parameters

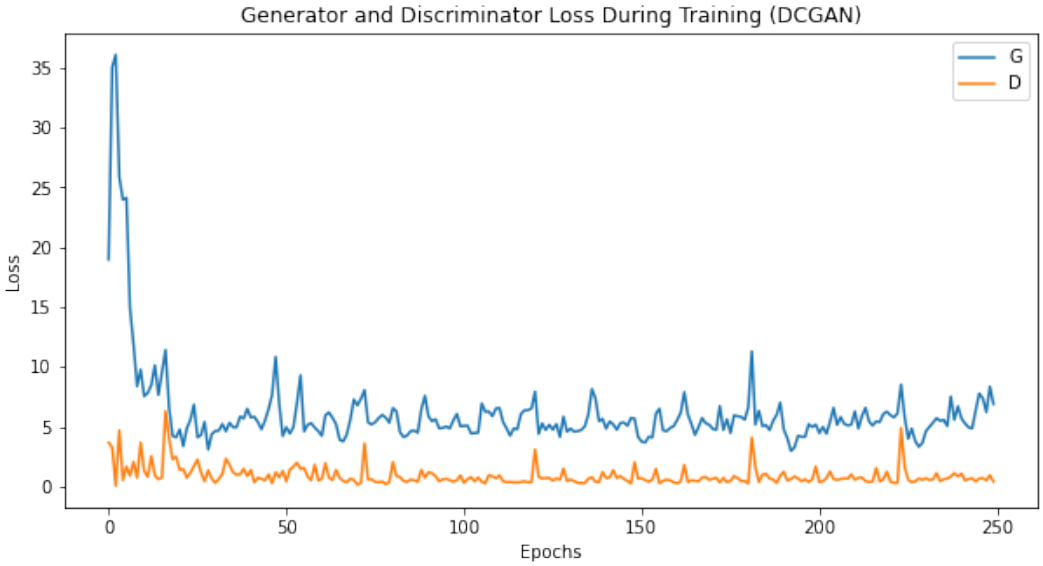


Figure 12: DC-GANs Model Loss Plot for best Hyper-parameters

A detailed comparison of FID Score, Inception Score as well as training times can be observed in the Table 1 below.

Table 1: Comparison of Diffusion Model and DCGAN Model

Metric	Diffusion Model	DCGAN Model
Training Time (seconds)	955.84	1280.67
FID Score	1199.38	1125.79
Inception Score	1.65 ± 0.05	1.83 ± 0.23

A comparison of images generated by each of the two models can be observed in the Figure 13 below.

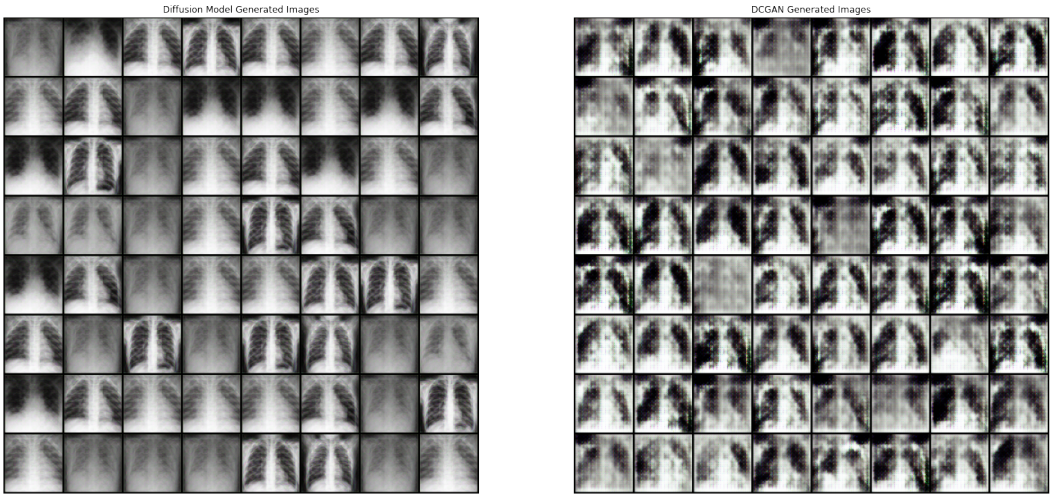


Figure 13: DC-GANs Model Loss Plot for best Hyper-parameters

6 Conclusion

This study used and analyzed two cutting-edge generative models, the U-Net-based Diffusion Model and the Deep Convolutional GAN (DCGAN), to synthesize COVID-19 lung images. The major goal was to compare their performance in terms of training efficiency and image quality, which would provide insight into their different strengths and flaws.

The Diffusion Model and the DCGAN were trained on the same dataset, and their performances were evaluated using the Frechet Inception Distance (FID) and the Inception Score. The training times of both models were recorded to assess their computational efficiency.

The Diffusion Model had a shorter training time of 955.84 seconds than the DCGAN's training time of 1280.67 seconds, indicating its computational efficiency. This performance can be due to the Diffusion Model's iterative denoising process, which takes advantage of the structured U-Net design.

The DCGAN had a lower FID score of 1125.79 than the Diffusion Model of 1199.38, indicating that the images generated by the DCGAN were more similar to the original images in terms of feature distribution. Both models had quite high FID scores, indicating that there is opportunity for development in terms of generating more realistic images.

The Inception Score reinforced the quantitative assessment, with the DCGAN receiving a higher score of 1.83 ± 0.23 than of Diffusion model's 1.65 ± 0.05. This shows that the DCGAN generated more diversified images with more discernible lung features than the Diffusion Model.

Although the quantitative measurements favored the DCGAN marginally, a visual examination of the generated images revealed that the Diffusion Model provided images that were considerably closer to the original lung scans. As seen in the comparison Figure 13, the Diffusion Model successfully denoised the photos to a high level of realism, capturing finer features and structures than the DCGAN. In contrast, the DCGAN-generated images frequently contained artifacts and poorly defined features.

The U-Net-based Diffusion Model revealed consistent training dynamics, thanks to the organized and hierarchical character of the U-Net design. In comparison, the DCGAN, while capable of creating higher quality images based on metrics, required more precise hyperparameter adjustment to accomplish consistent training.

7 Future Work

Future work could focus on several areas to enhance the performance of both models:

- **Hyperparameter Optimization:** Fine-tuning hyperparameters such as learning rates, batch sizes, and noise levels may improve the training stability and image quality in both models.
- **Architectural Enhancements:** Investigating sophisticated architectures, such as conditional GANs or attention mechanisms, could improve picture synthesis performance.
- **Data Augmentation:** Increasing the diversity of the training dataset using advanced data augmentation techniques may help the models develop more robust representations, resulting in better generalization and image quality.

8 Implementation

In this project, both group members worked together to create the majority of the coding, assuring originality and following best practices. The diffusion model was developed using the methodology described in the original research paper, assuring a correct translation of the presented techniques into code. The PyTorch documentation included an example that led the implementation of DCGANs. Although the main architecture and training loop were adapted, we made major changes and tweaks to meet our specific project requirements. We used PyTorch documentation for advice on data loading and augmentation, however we wrote our own code to handle and preprocess our dataset. To properly measure the quality of created images, the computations for FID (Fréchet Inception Distance) and Inception Score were developed from scratch based on theoretical knowledge. Custom code blocks were used to construct all outcome visualizations, such as graphs and image comparisons. We secured the originality and integrity of our work by writing the majority of the code on our own and only used references to match with known procedures.

9 Supplementary Material

9.1 GitHub Repository

https://github.com/ParamChordiya/ECE-285-Intro-to-visual-learning-Project.git

10 Project Contribution

10.1 Param Chordiya

- **Coding and Development**: Responsible for the implementation of U-Net based Diffusion Models.
- **Evaluation Metrics**: Developed the classes for calculating the FID score and Inception score.

10.2 Riya Joshi

- **Coding and Development**: Implemented the DCGAN, including the generator and discriminator networks.
- **Visualization**: Created functions for plotting the results.

10.3 Report Writing

- **Equal Contribution**: Both members contributed equally to the writing of the report and presentation.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[2] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[3] Dong Nie, Roger Trullo, Jun Lian, Caroline Petitjean, Su Ruan, Qian Wang, and Dinggang Shen. Medical image synthesis with context-aware generative adversarial networks. *Medical Image Computing and Computer Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part III 20*, pages 417–425, 2017.

[4] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 289–293. IEEE, 2018.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[6] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[7] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[8] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[10] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[11] Pranav Raikokte. Covid-19 image dataset, 2020.

[12] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, et al. U-net: deep learning for cell counting, detection, and morphometry. *Nature methods*, 16(1):67–70, 2019.