Fictional Face Generation using DCGAN and WGAN with Gradient Penalty

MSc Research Project
MSc In Data Analytics

Yogaraj Kori
Student ID: x23241365

School of Computing
National College of Ireland

Supervisor: Abid Yaqoob

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	……. …………………………Yogaraj Kori……………………………………………………………
Student ID:	…………………………………………x23241365…………………………………………….……
Programme:	………………MSc Data Analytics………….. **Year:** ……………2024…….
Module:	……………………………Research Project……………………………………………….………
Supervisor:	……………………………………Abid Yaqoob…………………………………………….………
Submission Due Date:	………………………………12/12/2024……………………………………………………….………
Project Title:	Fictional Face Generation using DCGAN and WGAN with Gradient Penalty
Word Count:	…………7533…………………… **Page Count**……………..23…………………….…….

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……………………………………Yogaraj Kori………………………………………………

Date: ………………………………………………12/12/2024…………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Fictional Face Generation using DCGAN and WGAN with Gradient Penalty

Yograj Kori
X23241365

Abstract

With the increase in the capability of Artificial Intelligence and Machine Learning Algorithms, scams and deep-fake profiles on social media websites are becoming predominant. As such it is important to counter such malicious activities by bettering existing human face detection algorithms and testing them with deep-fake images of our own. In this project, we will be providing **a human face generation algorithm** with the ability to create high end facial images of non-existent personalities using **Deep Convolutional Generative Adversarial Networks (DC-GAN) and Wasserstein Generative Adversarial Networks with Gradient Penalty (WGAN + GP)**. A collection of thirty thousand images of cropped and augmented human faces was used to train the model where two parts of the algorithm generator and discriminator compete with each other in creation and challenging of new images. To analyze the model performance, we will be monitoring **Inception Score (IS)** of the generated images along with plotting the generator and the discriminator loses to understand the imbalance. Along with Inception Score we have also utilized **Structural Similarity Index** (SSIM) which compares the contextual and structural similarity of generated images and its training image counterparts as well **as Fréchet Inception Distance (FID).** With this implementation we can generate non-existent human faces and utilize it to further improve facial recognition software by feeding these deep-fake images for negative validations and edge case scenario validation.

Keywords: Deep Convolution Generative Adversarial Network (DC-GAN), Wasserstein Generative Adversarial Networks + Gradient Penalty (WGAN + GP), Inception Score (IS), Fréchet Inception Distance (FID), Structural Similarity Index (SSIM), Negative validation, Edge Case validations

1 Introduction

While the framework for Generative Adversarial Networks was introduced in 2014, it still serves as an exciting topic and several models resembling the original idea of two different neural networks competing and improving each other have been developed. While such models can be considered semi-supervised models, the purview of **GANs can fall under both semi-supervised and unsupervised learning**. The conceptual idea behind the model is similar to a author and a plagiarism model, where the author creates contents based on his own creativity and the plagiarism model compares the authors writing to its own understanding of other existing works and gives its opinion of whether the content is self-written or plagiarized. In this example although the work of author and the model might seem reliant on each other, they work concurrently creating and challenging contents.

From the example above, we understand that GANs have a neural network which exclusively generates contents which is inherently called **Generator** and an adversarial neural network which challenges the Generators content called the **Discriminator**. The Generators G Generatively do not have access to the training data and immerse themselves in creating content which in the case of our project are images. While the Discriminator D has access to the dataset, it compares the image generated by the Generator G with the training data and gives a decision as to whether it is a fake image or a real image. If the Discriminator judges an image as fake, the Generator works harder on creating more realistic images, similarly the Discriminator keeps improving itself in determining real and fake images until a point where it is hard for it to determine and it plateaus.

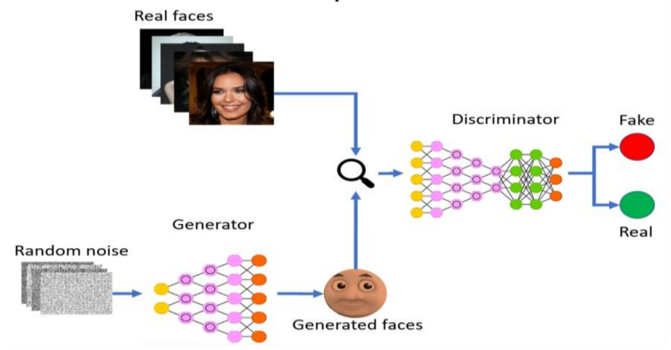


Fig 1: Working of GAN.

The model is better explained with the figure 1 where the **Generator utilizes random uniform noise** and generates human faces and the **Discriminator model which has access to training images** determines whether the image is real or fake. Each error made by the Discriminator is backpropagated to the Generator. Generator is further trained with random high dimensional noise called the **latent space**. This function of Generator is mathematically defined as **G: G(z).** The Discriminator can be viewed as a probability matrix of 0 and 1 , 0 being the image being fake and 1 being the image being realistic. For a simple GAN we can denote the Discriminator D as **D: D(x) → (0, 1).** When the generator's output distribution perfectly matches the real data, the discriminator becomes unable to differentiate between real and fake images, predicting a probability of 0.5 for all inputs. This equilibrium represents the ideal state of a GAN, although achieving this balance is often challenging during training.

Deep Convolutional Generative Adversarial Networks are a type of GAN which implements GAN utilizing the convolutional powers for better image generation. It leverages the deep convolutional Neural Networks to generate realistic images. It was first introduced by Radford, Metz and Chintala in 2015, where the key concept was to utilize the powers of both GAN and CNN for both the generator and the discriminator. In this implementation generators learned to map the latent space to the data space of the real dataset, while the discriminator acts as a challenger to the generator evaluating its performance. These models incorporate architectural features like **strided convolutions, ReLU and Leaky ReLU activations** along with tanh activation models and batch normalizations to generate images. The other positive note for this model was the stable training of the model and an enhanced output quality of the images. Along with image generation they are also used in style transfers and unsupervised feature learning.

Wasserstein Generative Adversarial Network with Gradient penalty is another such model which was improved upon the initial GAN and the WGAN first introduced by Arjovsky et al. It leverages the **Wasserstein distance** for comparing the real and generated images providing more meaningful loss values while also improving the generators feedback. But the WGAN first implemented used **Lipschitz constraint** by weight clipping which led to a limited model capacity and convergence of generator and discriminator model outputs. To overcome this a **gradient penalty** was introduced which penalizes the model when the **normal deviation grows or drops below 1** ensuring the Lipschitz constraint. This implementation works similarly to DCGAN, however when the generator output varies from the gradient norm, it adds a penalty to the discriminator loss which ensures better quality images as the model progresses its training. Not only can this model be utilized for image generation, but it can also be utilized in image synthesis, 3D models creation and audio generations. It is not only more stable than a typical DCGAN, it is also faster in computing and produces images of a higher quality than that of DCGAN.

GANs have numerous applications across various domains. For example, in the fashion industry, GANs can be used to create images of virtual models, reducing the need for costly photoshoots and logistical expenses. This allows for the **creation of diverse advertising campaigns**, potentially enhancing customer engagement and sales. Additionally, after adequate training, GANs can enhance image quality by preserving intricate details such as colour, background, and textures, making them valuable for tasks like image super-resolution, restoration, and content generation. These capabilities demonstrate GAN's potential to revolutionize industries that rely on high-quality and diverse visual content.

Although there are more recent variations of GAN available today like StyleGAN, we chose to implement DCGAN and WGAN_GP due to the resource intensive training of the models. **StyleGAN and other recent models require significant memory and computational power** along with extensive hyperparameter tuning for their implementation and since we will be implementing the models in a memory and **computational power constraints**, DCGAN and WGAN_GP would help us in answering the below Research questions significantly.

1.1 Research Problem/ Question

Q1: Can machine learning algorithms be used to generate accurate human faces and if so, how accurate are the generated images. Once we answer these questions, we will also answer how the model performs for each of the different factors mentioned above.

Q2: How does the training size factor into the performance of a complex model like Generative Adversarial Networks?

Q3: Does the resolution of images that the model is trained on affect the accuracy of the model.

Q4: How does the Generator and the Discriminator of the model perform throughout the model training?

Q5: Can we measure the accuracy of the generated Images using strategies like Inception Score or Structural similarity Measure.

With this project, we will be trying to answer these Research question which from hereon will be mentioned with their Q numbers.

2 Related Work

2.1 DCGAN

The authors of the paper [1] explore on the pose invariant image synthesis data to generate images for the model while training the model on Queen Mary University of London Multiview Face dataset. Generative Adversarial networks was implemented with Generator hyperparameters as learning rate 0.002, beta1 as 0.5, Adam as Generator optimizer similarly for the discriminator learning rate as 0.0002 and 0.5 for beta1 and like generator Adam as the optimiser. Although the model was trained using a Deep Convolutional Generative Adversarial Network with an Fréchet Inception Distance (FID) of 74.05, they were not able to generate a proper image for the model and achieved a dip in the curve in the FID score during the initial and end of the curve of the epochs, which indicate that the generator and the discriminator performed poorly during the start and the end of the training. Contrary to the performance to the [1], the authors of [2] use the Celeba dataset which provides higher amount of training images. In [2] the authors perform image generation using GAN but also perform image recognition to check the accuracy of the generated image, although this seems like the right approach the authors were not able to achieve greater accuracy at the same time. They used the model with dataset labeling which consists of over 40 binary attributes, including facial attributes (Bald, Blond hair, Black hair, Eyeglasses, Smiling, etc) , demographics, ethnicities, ages and genders.

While both papers give a wide variety of results and technical contributions the results were inconsistent. However, in [3] they implement DCGAN similarly to the above papers with the exception of utilizing one-class classification model to judge the accuracy of the facial images and the generated images. Along with the one-class classification model they also implemented various filter enhancement methods along with Multi Channel Convolutional Neural Networks (MCCNN). The generated images were then fed as inputs to a image classification model where the accuracy of the models were interpreted. The accuracy of the MCCNN was defined to be higher than the accuracy of the other industry standard models showing a great innovation, however the model was not trained on higher quality images and were implemented on limited number of image datasets and might not produce similar results under different circumstances. [4] proposes human face generation using unlabeled data and random noise They use models like DCGAN and a simple GAN. They also implement techniques like noise removal, image normalization and Histogram Equalization and Contrast enhancement. They also used techniques like data augmentation, data imputation and transferred learning with pretrained DCGAN, while these techniques increased the accuracy of the model and enhanced model performance, they were unable to achieve highly defined accuracy because the results were examined using the loss functions of a simple GAN where the generator generates images from the scratch and the discriminator tries to decide whether the image is fake or real. While the implementation seems relevant to the trends, they do not have proper evaluation metrics for the model as techniques like Structural Similarity index and Inception Score help with the proper model evaluation.

The authors [5] in this paper uses Deep convolutional generative adversarial networks to create fake face images. They have used CelebA dataset as the training dataset for the same. The design followed in this paper is the “generator- used to generate the fake face images and discriminator- used to differentiate between real face images and fake face images” method. The training is done with 40 epochs, the generator and the discriminator will be trained in turn; to calculate the losses, they have used loss functions that are previously used. This model is trained on GPU along with model inputs being moved to GPU. By the end of this research the model was able to generate fake human face images. Even though they were able to produce the results as same as human faces, their model was not stable enough as well the faces were shady because they were not clear in differentiating the skin color of the face with respect to white and black. Another important factor is the execution of the model for the lower epoch leading to an undervalued output. Similar to this, the [6] propose a method to generate fake human face images using Deep Convolutional Generative Adversarial Network (DCGAN) technique. And use Measuring the Quality of the Features of an Image (MEQFI) to evaluate the images produced. They used three datasets in this model as the training dataset namely CelebA, CelebA-HQ3 and Labelled Faces in the

Wild LFW dataset. The methods used for the model development were batch size = 128, image size =64×64, noise vector =100, number of features in both of the Generator and the Discriminator 64, and they used the Adam optimizer [14] to update both G and D neural networks with a learning rate of 0.0002 and β1 =0.5. They have managed to generate fake face images with better quality and with the help of good dataset examples, but as mentioned in the paper their model could not be as stable as they have expected it to be because of training process used in DCGAN and they can achieve more score from the measure if they follow more effective processes.

The authors [7] proposed a model using Pooling layers, DCGAN, Network Architecture, Hyperparameters, Training processes and evaluation parameter. So, after the series of implementations and simulations, they could generate the fake face images from the DCGAN model which they got as the low-fidelity 64*64-dimension images. They were able to generate the artificial face images which had the structural similarity of up-to .34 with respect to the original dataset of CelebA training data. But this model can be improved to get a high-resolution image with better hardware capacity and improved GANs techniques and processes. Another issue with the model was that no further preprocessing on the input images. Unlike this paper, in the next paper the [8] proposes a model to reconstruct the personal image from forensic sketches using DCGAN. They have used CHUK Faces as the dataset. The proposed design was developed representing the model-specific function stage performed with a specific Pre-processing stage- This stage consists of noise reduction, image resizing and normalization for consistency and improve the input quality for the later stages. By the end of the processes and techniques followed, using the DCGAN algorithm they managed to reconstruct the personal face images by transforming the sketches into vibrant realistic images. The image quality was evaluated by the nearest neighbor method. Although this paper utilized great process implementation, The evaluated score was not up-to the mark.

2.2 WGAN+GP

The main issue with the DCGAN is its instability in training, which is harder to cure, when either the generator or the discriminator starts behaving badly, we have to ensure that it is corrected. One such implementation of GAN called the Wasserstein GAN with gradient penalty addresses this issue by applying a penalty on the generator when the model performs poorly.[9] talks about the implementation of WGAN_GP where the model was trained on human faces aiming to utilize GAN for facial reconstruction and medical engineering. The positive note of this implementation is the utilization of small dataset and achieving higher accurate images. They utilize Wasserstein loss as a way to improve the losses from both the generator and the discriminator leading to a more stable model training. But the negative affects of this implementation is that model stability is not focused on, while the model does perform better at initial stages, it does not however have a steady model training and the discriminator overwhelms the generator after certain epoch.

[10] talks about the implementation of multiple variations of GAN including WGAN+GP, where the authors of the paper implement and compare the model performances for 128 X 128 pixeled images, The generated image resembles real human beings, however, doesn't reach the level of which the original paper demonstrated because of insufficient computing power and limited running time. The paper also talks about the ethical concerns of the Human Face Generation using GANs and makes a prospect for the future of the human face generation. They also found that WGAN with weight clipping overlooks data

distribution's high-order moments leading to extra layers of the input dataset images being clipped from its source and the model is trained on clipped images leading to a higher quality Human face generation. The results from this model in the paper demonstrated higher accuracy of human faces with decreased deformities, however the margin blurred depiction still remained a problem for the images.

3 Research Methodology

This research was implemented using a dataset with a moderate sized unlabelled image. While GANs require huge datasets with labelled images for better image generation, we are trying to improve the accuracy of the GAN by using an unlabelled dataset. With this we will be following the traditional approach to GAN as mentioned in the figure 2 and we will be **experimenting with different approaches including activation functions, hyper-parameters, batch sizes and image resolutions.** The model performance also hinges on the number of epochs the generator and the discriminator are trained on.

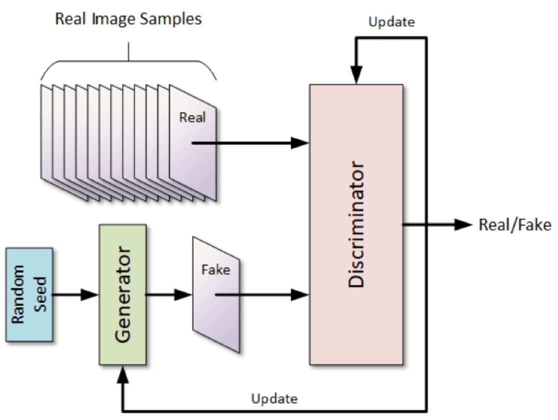


Fig 2: Framework of GAN utilized.

3.1. Libraries Utilized

Since Image generation at first involves the preprocessing and augmentation of images, we will be utilizing the **TensorFlow library to process the images**, and we will also be **saving the images into tf.records format** since image processing is a time and labour-intensive process. Since we will also be processing the data, we will be **requiring CV2 library to process the data**. The pre-processed images are stored in the form of a four-dimensional array **using keras preprocessing library**. The next step is to implement the GAN, for which we will be utilizing convolutional neural networks using keras for both generators and discriminators.

3.2 Data Gathering

One of the most popular datasets used in Image processing and the image generation **sources is the CelebA website** which provides a vast dataset of cropped images of celebrities. The original dataset has 202,599 images of various celebrities of different skin colour and different facial structures. The images are also taken from different time frames and different backgrounds, but all match a similar structure and are aligned and cropped in the dataset. Although the dataset contains massive amounts of images, for the purpose of our project we will be taking only 30,000 images which will reduce the training time and the computational complexity of the model.

Fig 3: Dataset Images.

3.3 DATA AUGMENTATION

In this step, we will need to increase the complexity of the images dataset so the model can perform better and can be trained for vast set of heterogeneous data. As such we will be performing **Data augmentation** on the dataset, where we will take existing images and rotate the images at different angles leading to a more diverse training of the model. In most of the cases, the images have been rotated **at an angle of 30 degrees** either in clockwise direction or anti clockwise direction and in some cases the images have been completely flipped upside down, so that the model can be trained on a more diverse dataset.

3.4 DATA Preprocessing

This is one the most important and time consuming process in the project, as we are dealing with image data, we will need to process all the images along with the image path, For the purpose of better utilization of time during the model building, we have the image dataset in a Tensor Flow record format which can be easily ingested by both the Generator and the Discriminator. Since the original dataset has copious amounts of images, we will be **utilizing the Kaggle API** to directly download and unzip the images from the API. Once images are available, we will be converting **the images back to an RGB format**, following which we will be able to convert it into **a 4-Dimensional NumPy array**, but the most important step is to **convert the images in a latent space of [1, -1].** This is a crucial step in our preprocessing as the model can not recognize any input exceeding the range of this latent space and ensures that all the features of the image contribute equally to the model training.

As seen in figure 2, the loss of the discriminator is looped back to the generator model so that the generator performs better. as such we have utilized **cross entropy model loss calculations** for each generator and the discriminator where generator loss is calculated as the cross entropy of the fake output whereas the model loss for the discriminator is calculated as the cross entropy comparison of the fake output of the generator to the fake output of the discriminator, similarly the real loss of the model is calculated as the entropy comparison of the real output and the real images. the final model loss for the discriminator is calculated as the sum of real loss and the fake loss.

3.4 Model building

3.4.1 NOISE GENERATION

The first step towards the model development is the creation of noise array also called **the latent dimension array**, which is an array of normalized random values between the space of [1, -1]. The below figure shows us the Generated Latent space noise image which has been normalized.

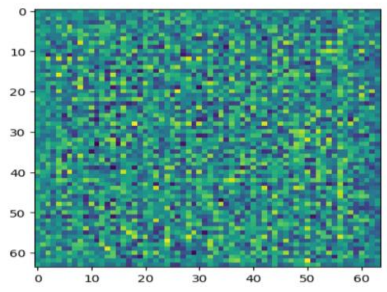


Fig 4: Noise Generation.

3.4.2 GENERATOR

Now that we have a clean, pre-processed and augmented dataset which has been converted into a NumPy array with a latent space of [1, -1], we will now be working on building on building the Generator and the Discriminator for the model. The Generator G does not have access to the input numpy array and as such will need to have random uniform noise as its input. The number of layers in the Generator will depend on the pixelation of the images we intend to generate but for the purpose of this project, we will be generating images of 64 X 64. The generator model contains **multiple layers of 2D Convolutional Neural Networks**, where the initial layers work on generating the images from higher image resolutions and the later layers work on fine tuning the lower resolutions of the image which can be seen as the initial layers looking at the bigger picture while the later layers work on fine tuning smaller and more intensive parts of the images. The figure 5 shows us a better understanding of the development of the Generator model.

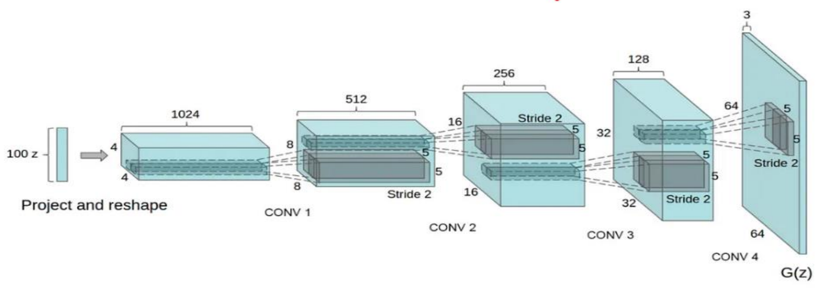
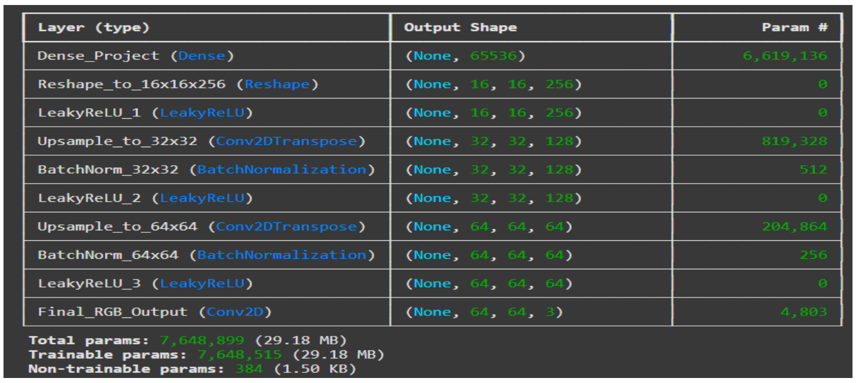


Fig 5: Generator Layers.

For the project, we will be utilizing **he_normal kernel initializer** which will help in faster initialization of the generator's first layer following which the last layer of the model will have tanh activation function. The figure 6 shows the summary of the Generator function once it is built and trained.



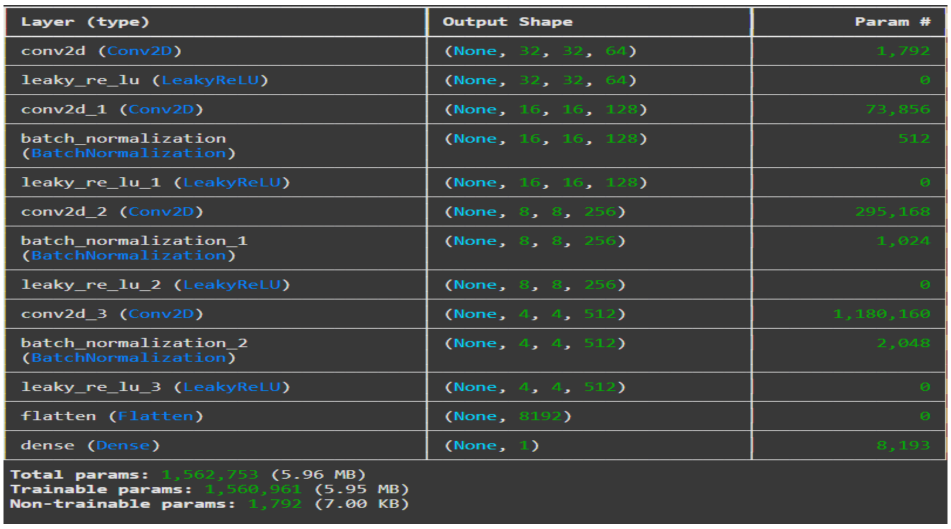
Layer (type)	Output Shape	Param #
Dense_Project (Dense)	(None, 65536)	6,619,136
Reshape_to_16x16x256 (Reshape)	(None, 16, 16, 256)	0
LeakyReLU_1 (LeakyReLU)	(None, 16, 16, 256)	0
Upsample_to_32x32 (Conv2DTranspose)	(None, 32, 32, 128)	819,328
BatchNorm_32x32 (BatchNormalization)	(None, 32, 32, 128)	512
LeakyReLU_2 (LeakyReLU)	(None, 32, 32, 128)	0
Upsample_to_64x64 (Conv2DTranspose)	(None, 64, 64, 64)	204,864
BatchNorm_64x64 (BatchNormalization)	(None, 64, 64, 64)	256
LeakyReLU_3 (LeakyReLU)	(None, 64, 64, 64)	0
Final_RGB_Output (Conv2D)	(None, 64, 64, 3)	4,803

```
Total params: 7,648,899 (29.18 MB)
Trainable params: 7,648,515 (29.18 MB)
Non-trainable params: 384 (1.50 KB)
```

Fig 6: Generator summary.

3.4.3 DISCRIMINATOR

While the generator is trained on a random normalized noise in the latent space, the **discriminator is trained on the images array**, following which the model can decide on whether the generated image is real or fake. The discriminator works inversely to generator as the model seeks for the lower resolution of images first and then with the later layers the model starts looking at the higher resolutions of the images. This is implemented like the generator and the summary is represented below.



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 64)	1,792
leaky_re_lu (LeakyReLU)	(None, 32, 32, 64)	0
conv2d_1 (Conv2D)	(None, 16, 16, 128)	73,856
batch_normalization (BatchNormalization)	(None, 16, 16, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 16, 16, 128)	0
conv2d_2 (Conv2D)	(None, 8, 8, 256)	295,168
batch_normalization_1 (BatchNormalization)	(None, 8, 8, 256)	1,024
leaky_re_lu_2 (LeakyReLU)	(None, 8, 8, 256)	0
conv2d_3 (Conv2D)	(None, 4, 4, 512)	1,180,160
batch_normalization_2 (BatchNormalization)	(None, 4, 4, 512)	2,048
leaky_re_lu_3 (LeakyReLU)	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 1)	8,193

```
Total params: 1,562,753 (5.96 MB)
Trainable params: 1,560,961 (5.95 MB)
Non-trainable params: 1,792 (7.00 KB)
```

Fig 7: Discriminator summary.

3.4.4 LOSES

Loses in GAN are used by both the generator and the discriminators to create realistic images, where the generator uses the Wasserstein Loss to trick the discriminator, and the discriminator uses it to understand the reality of the generator output. Tracking the loses for both of these provide valuable information on training of the model. Ideally both the generator and the discriminator losses need to converge as the model learns better which implies that the generator is creating exceptional images and the discriminator is unable to prove that the images are fake.

3.5 DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

With the implementation of generator and the discriminator shown above we were able to implement DCGAN. This is a Generative Adversarial Network architecture which involves replacing the pooling layers with strided convolutions for discriminator and strided fractional convolutions for the generator. It also involves **using batchnorm and removing hidden layers** which are fully connected to allow us to modify the model for deeper architectures. This also allows us to **implement LeakyReLU** for all the layers in our discriminator. This can be considered as an evolved version of our simple GAN and as such **produces higher resolution images** which are more well defined, since Convolutional Neural Networks greatly enhances the generated image quality. The main issue regarding this model is the tracking of the model performance leading to a **confused model development** along with the **disruption in the model stability**. To improve on this, we have also implemented a different model which can perform more stably.

3.6 WASSERSTEIN GENERATIVE ADVERSARIAL NETWORKS WITH GRADIENT PENALTY

Along with DCGAN, we were also able to implement Wasserstein GAN which involves loss formulation called the Wasserstein loss along with a **Gradient Norm penalty** every time the model performs poorly. This model using **gradient clipping using weights** to achieve model correction at any given moment of the training. The main issue with this model however is **the potential of gradient explosion or a gradient vanishing** which happens when the model is unable to decide on a model weight and keeps increasing the model weight so the generator and the discriminator starts performing worse and worse leading to a model disruption. However, with the help of Gradient Penalty we can overcome this result where whenever the model performs irregularly, we can punish the model into performing better leading to a more stable model generation.

4 Implementation

4.1 Implementation Requirements.

Image generation is a hardware intensive process requiring high Functioning **Graphical Processing Units and high RAM**, as such for the purpose of project development we have utilized Google collab to handle GPU requirements. We have utilized a System RAM of 15 GB as well as GPU RAM of 15 GB with 80% to 90% of the RAM utilization. These configurations help in the model training allowing the model to perform efficiently and without delays.

4.2 Model Implementation

Our main objective of the project was to answer the Research Questions mentioned in the 1.1 section of the report along with implementation of two different models which behave differently in two distinct methods. So that we will be able to **compare and contrast not only the model performance** but also how the steps prior to the model training affects the performance of the model. The below subtopics not only highlight the model development but also define the structure of implementation as well as define the best hyperparameters for the models and also includes the best evaluation metrics for each of the models. After multiple executions, we were able to define the best configurations for the model to perform with the highest efficiency. As such we have been able to define the best configurations as below.

4.2.1 DCGAN

We began our project with the implementation of this model and after weighing in the computational costs and the execution time of each epoch of the model, we were able to find the best optimal flow for the project is shown in figure 8.

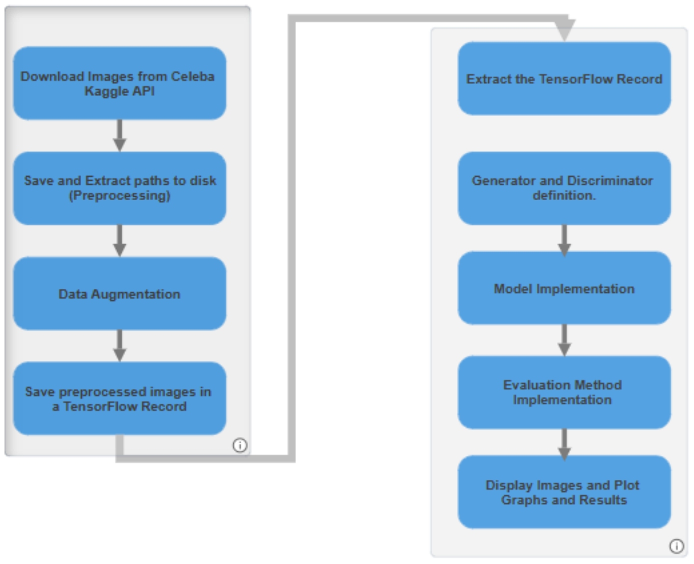

Download Images from Celeba Kaggle API
Save and Extract paths to disk (Preprocessing)
Data Augmentation
Save preprocessed images in a TensorFlow Record
Extract the TensorFlow Record
Generator and Discriminator definition.
Model Implementation
Evaluation Method Implementation
Display Images and Plot Graphs and Results

Fig 8: WGAN_GP implementation Flowchart.

We used **data preprocessing as a separate step** in the model development rather than as a model implementation step since the major consumption of time in the model execution was taken up by **the preprocessing and augmentation of the images**. As such we downloaded, pre-processed and augmented images from the dataset and saved the pre-processed data as a TensorFlow Record. Following which we implemented our model by first extracting the TensorFlow Record and then building the generator and the discriminator followed by model implementation and then display of the result and then calculating the evaluation metrics for the generated images. Because **DCGAN is a slower model** when compared to other models like WGAN_GP, we have utilized **a training size of only 3000** Images with an **image resolution of 64 X 64** both of which were deduced from our research for the most optimal execution while considering the constraints of our own computational capabilities both physical and logical.

Upon implementation of our initial model with fewer layers, we then improved on it but adding more layers to the model. As such we were able to summarize the number of layers required for both the generator and the discriminator and its optimal execution. The table 1 explains the number of layers in the model.

Layer Name	Number of Generator layers	Number of Discriminator Layers
Dense Layer	1	1
Reshape Layer	1	0
LeakyRelu Layer	3	4
Conv2DTranspose Layer	3	4
Batch Normalization Layer	2	0

Flatten Layer	0	1
Input Layer	0	1

Table 1: DCGAN Layers.

Along with these layers, we also have hyperparameters which are set as follows

Hyperparameter Name	Value
Optimiser	ADAM
Learning Rate	0.0002
Beta_1	0.5
Beta_2	0.5
Epoch	100
Batch Size	32
Latent Dim	100
Input Size	30,000

Table 2: DCGAN Hyperparameters.

4.2.2 WGAN + GP

We were able to implement this model by utilizing the images from the Celeba dataset by downloading the dataset from the Kaggle API and directly loading the data into the google collab disk, following which we were able to generate, save the image paths in the disk and perform data augmentation on the images as a step in the preprocessing of the training dataset. For the purpose of implementing this model under both physical and logical constraints like the GPU requirements, model enhancements and evaluation capabilities of existing metrics, we have decided the optimal **training size to be 30,000 images** which we will vary after initial implementation to answer of Research Question Q2. For the same constraints explained above, we will able be implementing the models in order to produce images which are only **64 X 64 pixels** which will again be varied in order to answer the Research Question Q3. The whole model implementation can be seen in the flowchart in figure 9.

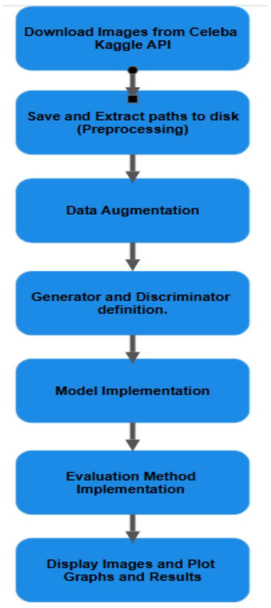

Download Images from Celeba Kaggle API
Save and Extract paths to disk (Preprocessing)
Data Augmentation
Generator and Discriminator definition.
Model Implementation
Evaluation Method Implementation
Display Images and Plot Graphs and Results

Fig 9: WGAN_GP implementation Flowchart.

Once the model was implemented, through our research we were able to quantify the layers necessary for both generator and the discriminator for the optimal execution. As such the Below table explains the number of layers for this model.

Layer Name	**Number of Generator layers**	**Number of Discriminator Layers**
Dense Layer	1	1
Reshape Layer	1	0
LeakyRelu Layer	4	4
Conv2DTranspose Layer	4	4
Batch Normalization Layer	4	0
Flatten Layer	0	1
Input Layer	1	1
Dropout Layer	1	1

Table 3: WGAN+GP Layers.

Along with these layers, we also have hyperparameters which are set as follows

Hyperparameter Name	Value
Optimiser	ADAM
Learning Rate	0.0001
Beta_1	0.5
Beta_2	0.9
Epoch	200
Batch Size	128
Latent Dim	128

Input Size	30,000

Table 4: WGAN+GP Hyperparameters.

4.3 Difference in EPOCH

We can see the difference between DCGAN and WGAN_GP in terms of epoch, this was implemented so as to check for the variation in the output levels of the model under different epochs. Since both the models are implemented with the same basic learning rates and other hyperparameters modifying the epochs is crucial in understanding the difference between outputs of each of the models.

5 Results

In this section, we will not only look at the images generated by both of our models, but we will also try to answer the Research questions mentioned in the section 1.1.

5.1 DCGAN

This was the first implementation of our model, and the model was trained on 3000 images from the Celeba dataset. We will be visualizing the generated images for this individually as this model has lower number of layers but also takes up huge amounts of computational power and time. First let us look at the dataset image on which the model will be trained on.

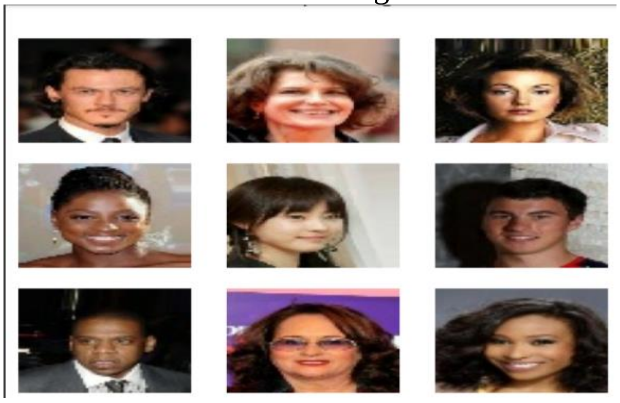


Fig 10: dataset images for DCGAN

Although we have visualized the dataset images in a grid, we will be training the model on individual images and **visualizing generated images sequentially**. The next image shows the first image generated by the model which will serve as a blueprint for the model to evolve on.

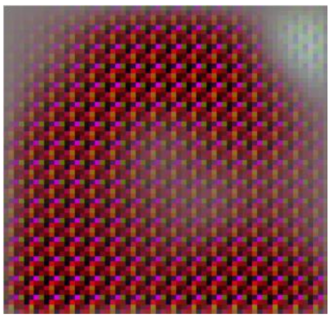


Fig 11: First image output for DCGAN

The first image of the model although might not look good does **show the basic structure of the images** which will be generated in further epochs. Now let us look at the final **output of the model after 100 epochs**.

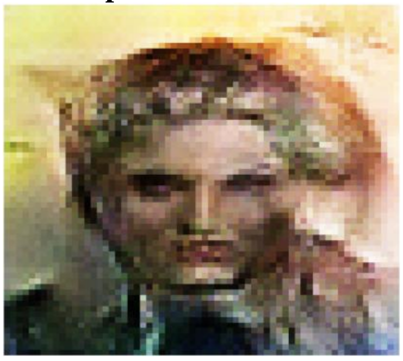


Fig 12: last image output for DCGAN after 100 epochs.

We can see that although the images generated **have a proper facial structure defined**, the image in itself **is not high quality and still has noise in it**. This maybe **a result of model instability** leading to either the generator or the discriminator overpowering its adversary. When the generator loss is high, it means that the model is stuck creating the same quality images and the discriminator keeps on rejecting the images proving they are fake, similarly when the discriminator losses are high, it means that the generator is producing more and more images of better quality that the discriminator is not able to keep up with the generator. However, since both the models are adversarial to each other, the discriminator cannot give proper feedback to the generator which makes the generator think that it is creating proper images even though it is not. The model performs better when both the generator and the discriminator produce meaningful losses. To counter the issue faced in this model, we will be developing another model called WGAN+GP where there is a penalty if there is model instability.

5.2 WGAN+GP

We have already established all the parameters for the executions in the above sections but let us first look at the dataset images for this model. Unlike the DCGAN, we will be **visualizing and generating the images for this model as a grid of 9 images** all being generated at the same time so that we can save time and computational resources for the model. This technique helps in visualizing and comparing the images efficient for us as well as saves computational powers and time and also helps in simplified visualization of the images. First let us look at the input images for the dataset as these images will be the training data upon which the model results are dependent on.

Fig 13: Input image for WGAN_GP

Now that we know the clarity and the quality of the images from the dataset let us look at the first generated image from the model, we have to understand that the model has not yet understood the dataset clearly and the gradient penalty to ensure the model performs better has not yet been founded by the model. This image will **serve as a blueprint for the model to generate better quality images**.

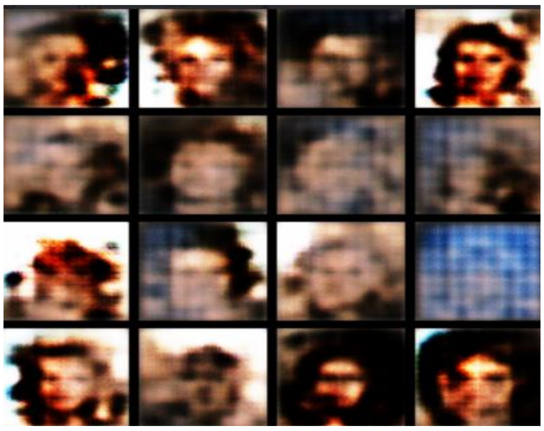
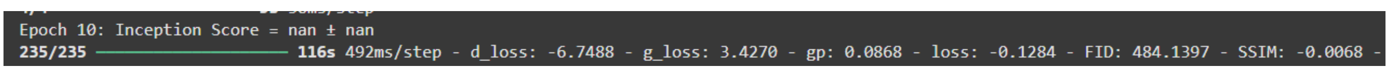


Fig 14: First generated image for WGAN_GP

We can see that while certain facial structures have been identified by the model in some cases, it has not yet figured out how to improve the model. Even on its first epoch the model generates a generator loss, a discriminator loss, a gradient penalty and an overall loss for the epoch. This can be seen in the image 15.



```
Epoch 10: Inception Score = nan ± nan
235/235 ━━━━━━━━━━━━━━━━━━━━ 116s 492ms/step - d_loss: -6.7488 - g_loss: 3.4270 - gp: 0.0868 - loss: -0.1284 - FID: 484.1397 - SSIM: -0.0068 -
```

Fig 15: Model results for an intermediary epoch.

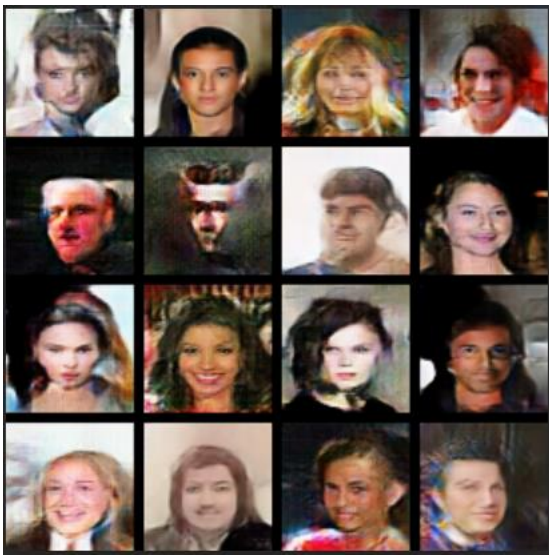


Fig 16: Final Image output after 200 epochs.

We can see that although **the quality of the generated images is not really high** when considering the generated image resolution to be 64 X 64, it is **creating more meaningful images than DCGAN**. The images not only show better quality in its essence it also **shows high similarity to a natural human being**. Thus, we can say that we have successfully been able to generate human faces with abstract personalities. The output of the model can be further enhanced by running the model on higher epochs from 200 to 500, better resolutions like 128 X 128 or 256 X 256 which can only be performed with higher computational efficiency. The final output of the model in the form of a gif can be viewed below.

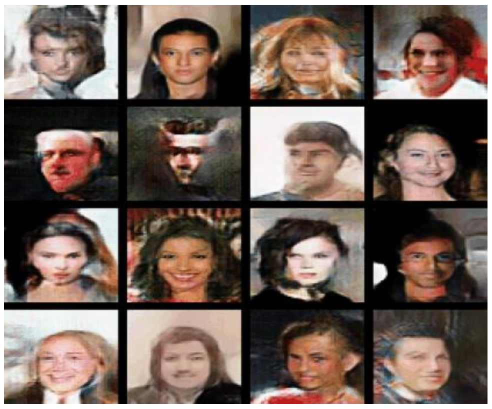


Fig 17: gif of final output.

Now let us look at **the generator and the discriminator losses** for the model and understand how the model is performing.

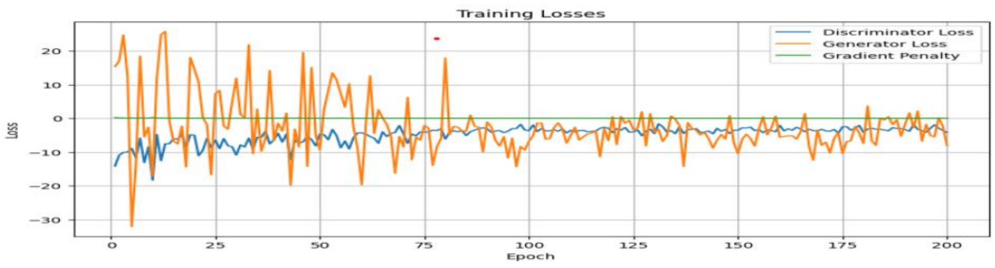


Fig 18: Generator and Discriminator behaviour

6 Evaluation

While there is still no accuracy metrics defined for a GAN human face generation, we can use metrics like Inception score, Structural similarity index and The Fréchet Inception Distance (FID) to properly evaluate the quality of the generated images from a GAN. Before we look at these metrics for the evaluation of the model, let us look at what each of these metrics calculate and what is the meaning of these metrics.

6.1 Inception Score

This is a popular metric for evaluating GAN's generated image quality. It **checks for the realism of the images** which can be defined as how lifelike the generated image is to reality. Another point of Inception Score (IS) is to **check for diversity of the generated image**. GAN models sometimes tend to pick out images from the training dataset which are favourable in producing images able to deceive the discriminator as such IS makes sure that the generated images are diverse. On the background, it utilizes a **pretrained model called InceptionV3**. The score is computed by the model analysing the generated images where it checks mainly for conditional label distribution and marginal distribution. A higher Inception score indicates that GAN has produced both realistic and diverse images. However, since IS utilizes a pre-trained model, it **lacks the ability to identify and calculate its score on the dataset specific features**. Now let us look at the inception score for both our models.

A. **DCGAN**

The below image shows the plot of the DCGAN Inception score as it evolves through the epochs. We can see that although the **generated image was not ideal**, the inception score keeps increasing although minutely as the model evolves which indicates that the model is creating better and better images which are both realistic and diverse according to the InceptionV3 model.

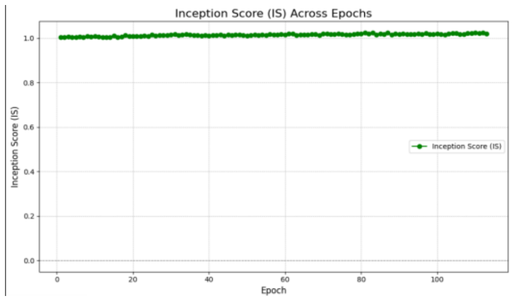


Fig 19: IS plot for DCGAN

B. **WGAN_GP**

The image 20 shows the plot of WGAN_GP's InceptionV3 model results. We can see that the values are much better than DCGAN. Although the model produced a well-rounded image at the end, the **inception score is still low**. However, we can see that as the model progresses that IS increases leading to a plot which is rising. This indicates that **our model is performing better at generating realistic images** and these images are diverse. Keeping in consideration that we trained the model with both physical and computational constraints we can safely say that this model has achieved higher accuracy than expected.

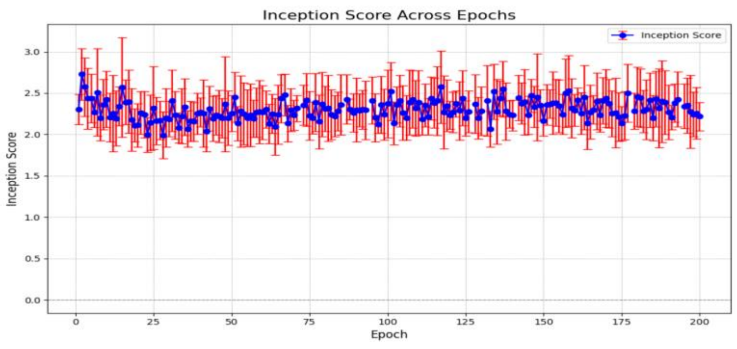


Fig 20: IS plot for WGAN_GP

6.2 Structural Similarity Index Measure

This is another popular metric used to measure the quality of the generated images by GAN. It checks for the **perceptual quality of the generated images**. It evaluates the generated images on **three basic factors luminance, contrast and structure.** It checks all three measures and combines them to present a score between -1 and 1 where -1 indicates an image with perceptual dissimilarity to a human face, while 1 represents a perfect similarity in the generated image and the image in the dataset. Since this does not depend on a pre-trained model like Inception Score, this is better for our evaluating our model. However, the SSIM can be **sensitive to changes in the brightness** and the generated image contrast and SSIM **does not check for the diversity in the generated image**. Now let us look at the SSIM score for both our models.

A. DCGAN

The image 21 shows the plot of SSIM score for the DCGAN model. We see that the majority of the results produce **SSIM score closer to -1** indicating that the model **does not present structural similarity to the input dataset**. However, the result might not be conclusive as indicated above SSIM score also relies on Brightness and contrast of the generated images. Considering this factor although we cannot overrule the SSIM scores for our model, we can state that our model has room for growth.

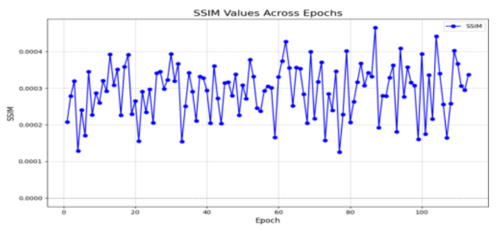


Fig 21: SSIM plot for DCGAN

B. WGAN_GP

The image 22 shows the SSIM scores for the model as it leaps through the epochs. We can see from the plot that the values are much higher than that of DCGAN and the **model performs reliably better than DCGAN**. We can also see that **SSIM score increases as the number of epochs** increases meaning that our model is generating images which have more structural similarities with the dataset images. Due to the disadvantages of SSIM mentioned above, the model might be performing better than the SSIM score is indicating. Considering all the mentioned factors and the SSIM scores calculated we can safely say that the model is performing satisfactorily and with the increase in the training dataset and the number of epochs the model will produce lifelike images of human faces.

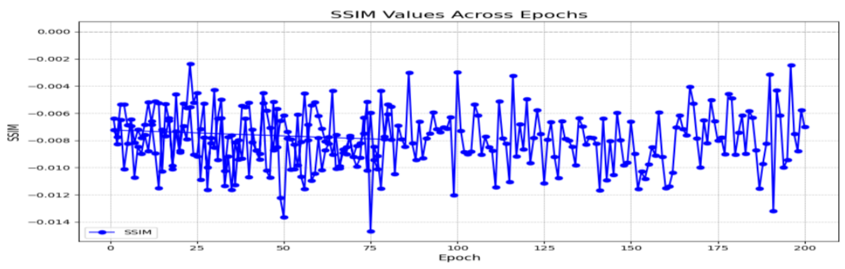


Fig 22: SSIM plot for WGAN_GP

6.3 Fréchet Inception Distance (FID)

The Fréchet Inception Distance is another popular metric used to evaluate the quality of generated images. It compares the **distribution of features between the dataset and generated images using the pre-trained InceptionV3 model**. It **calculates the mean and covariance** for the dataset and generated image which is then modelled as Gaussian Distribution. A higher FID score indicates that the generated image is not real nor diverse, similarly a lower FID score indicates that the generated image is both real and diverse. FID is comparatively better than IS even though both of them compare the same indicators as it takes into account both diversity and the quality of both the real and fake images which makes it sensitive to mode collapse and ensures the generated images are more closely resembling the real images. Similar to IS, FID also presents a sensitivity to dataset as it also works on the same pre-trained model and also requires large data samples for an accurate evaluation. Now let us look at the FID score of both datasets and compare both the models.

A. DCGAN

Although **the FID score seems low** for the model in the graph below, we can see that it evolves as the number of epochs increases. We can also see that the model has instability in training from the FID score as it fluctuates in the middle and since the image generated was not ideal the FID can be considered moderately adequate for our model considering we have only utilized 3000 images for training.

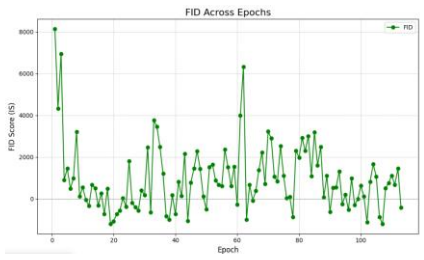


Fig 23: FID plot for DCGAN

B. WGAN_GP

The image 24 shows the plot of WGAN_GP's FID results. We can see that the values are much better than DCGAN. Although the model produced a well-rounded image at the end, the FID score is still low when compared to the baseline pretrained models. However, we can see that as the model progresses that **FID score increases** leading to a plot which is rising. This indicates that our model is performing better at generating realistic images and these images are diverse. Keeping in consideration that we trained the model with both physical and computational constraints and on a limited image dataset we can safely say that this model has performed Fairly.

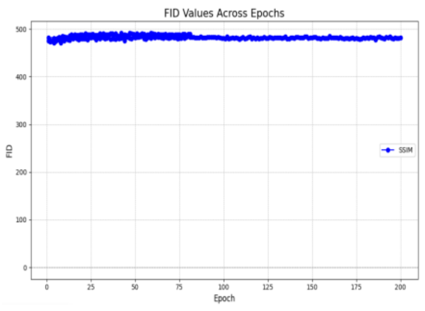


Fig 24: FID plot for WGAN_GP

7 Interpretation of Results

A. With the implementation of the models, we can safely say that with the help of GAN we can generate human faces of non-existent personalities accurately. Considering the results of WGAP_GP we can safely say that given the non-existence of computational complexities we can produce more realistic human faces which answers the question to **Research Question Q1**.

B. By comparing both the models performance and understanding that although several factors contributed to the accuracy of each of these models and the major one being the training size since DCGAN was trained on only 3000 images and performed poorly while WGAN_GP was trained 30,000 images and performed with more accuracy, we can safely **answer Research Question Q2** stating that training size does increase the performance of a GAN model given the computational requirements are answered.

C. Although this report was only including performances of 64 X 64 images trained GAN, upon increasing the resolution of training images the accuracy of the model also increases. This can be attributed to the fact that higher resolution images provide more fine-grained details for the generator and the discriminator to validly improve answering our third **Research Question Q3**.

D. We can see from the image, we can see that generator performs poorly at the beginning creating unrealistic and unclear images while the discriminator performs well at the beginning of the training, but as the training continues both the generator losses and the discriminator losses ideally should converge leading to a balanced state. Although it is highly unfeasible to produce an ideally stable model, from our implementation we can see that models try to converge at the centre answering our **Research Question Q4.**

E. From our evaluation metrics we can see that it is possible to measure the accuracy of the GAN generated human faces. Although each of the metrics has its own demerits we can measure the accuracy with great confidence. This answers our **Research Question Q5.**

8 Conclusion and Future Work

This project was implemented with the sole purpose of **generating human faces to be utilized for testing image recognition models** to check for spoofing in social media websites. Including answering the research questions stated, we have been **successful in generating realistic and diverse human faces of non-existent personalities** by utilizing the power of GANs and implementing DCGAN and WGAN_GP. We have also been able to determine that training size and resolution of the images greatly impact the quality of the generated images by the models. Not only did we monitor and visualize the behaviour of generator and the discriminator in an adversarial network, but we have also been able to track these behaviours for higher epochs. We have also been able to **measure the accuracy of the generated images by utilizing Inception Score, Fréchet Inception Distance and Structural Similarity Index.** Both by generating human faces using GAN and by measuring the generated image accuracy we have been able to successfully answer all the Research Questions presented at the beginning of the project.

By utilizing higher computational resources, we **will be able to implement better and more realistic human faces** which will confuse even facial recognition software across the globe. This will be essential in our war against spoofing and scams prevailing on the social media. Along with this we can also develop a text to image generation model where we can integrate Large Language models to help us generate images from text-based requests. These models can also be helpful in gaming industries and virtual reality industries saving vast amounts of money in these sectors. Additionally, we can implement domain expansion on the dataset ensuring that it involves face images of wide range of ethnicities, age groups and facial expressions.

References

Jagad Nabil Tuah Imanda, Bachtiar, F., & Achmad Ridok. (2023). Application of Deep Convolutional Generative Adversarial Networks to Generate Pose Invariant Facial Image Synthesis Data. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 7(5), 1049 - 1055. https://doi.org/10.29207/resti.v7i5.5112

Reddy, Shirisha. (2024). Unveiling Spoofing Attempts: A DCGAN-based Approach to Enhance Face Spoof Detection in Biometric Authentication.

Li S, Dutta V, He X, Matsumaru T. Deep Learning Based One-Class Detection System for Fake Faces Generated by GAN Network. Sensors. 2022; 22(20):7767. https://doi.org/10.3390/s22207767.

Generating Human Face with Dcgan and Gan. (2024). International Research Journal on Advanced Engineering Hub (IRJAEH), 2(05), 1348-1354. https://doi.org/10.47392/IRJAEH.2024.0186

Li S, Dutta V, He X, Matsumaru T. Deep Learning Based One-Class Detection System for Fake Faces Generated by GAN Network. Sensors. 2022; 22(20):7767. https://doi.org/10.3390/s22207767

Generating Human Face with Dcgan and Gan. (2024). International Research Journal on Advanced Engineering Hub (IRJAEH), 2(05), 1348-1354. https://doi.org/10.47392/IRJAEH.2024.0186

Phanindra, R.G., Raju, N.P., Vivek, T., Jyotsna, C. (2023). Face Model Generation Using Deep Learning. In: Choudrie, J., Mahalle, P., Perumal, T., Joshi, A. (eds) IOT with Smart Systems. Smart Innovation, Systems and Technologies, vol 312. Springer, Singapore. https://doi.org/10.1007/978-981-19-3575-6_20

Nekamiche, N., Zakaria, C., Bouchareb, S., Smaïli, K. (2022). A Deep Convolution Generative Adversarial Network for the Production of Images of Human Faces. In: Nguyen, N.T., Tran, T.K., Tukayev, U., Hong, TP., Trawiński, B., Szczerbicki, E. (eds) Intelligent Information and Database Systems. ACIIDS 2022. Lecture Notes in Computer Science(), vol 13757. Springer, Cham. https://doi.org/10.1007/978-3-031-21743-2_25

Junsuk Choe, Song Park, Kyungmin Kim, Joo Hyun Park, Dongseob Kim, Hyunjung Shim; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1940-1948

Hasan Sabah K. AL-Muttairi, Sefer Kurnaz, Abbas Fadhil Aljuboori,2024:Enhancing Cold Cases Forensic Identification with DCGAN-based Personal Image Reconstruction. **DOI:** https://doi.org/10.21123/bsj.2024.10896.

Hamdi, A., & Ghanem, B. (2019). IAN: Combining Generative Adversarial Networks for Imaginative Face Generation. *ArXiv, abs/1904.07916*.

H. Sengar, I. Nazir and S. Srinivasan, "Facial Detection and the Effect of Ageing," *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2023, pp. 666-671, doi: 10.1109/ICAC3N60023.2023.10541570.

T. Dong, "Effects of Different Generative Adversarial Networks on the Face Generation Task," *2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA)*, Changchun, China, 2023, pp. 1051-1055, doi: 10.1109/ICIPCA59209.2023.10257729.

Y. Xia, W. Zheng, Y. Wang, H. Yu, J. Dong and F. -Y. Wang, "Local and Global Perception Generative Adversarial Network for Facial Expression Synthesis," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1443-1452, March 2022, doi: 10.1109/TCSVT.2021.3074032.

Wei Wang, Mingwei Zhang, Ziwen Wu, Peiting Zhu, Yue Li,SCGAN: Semi-Centralized Generative Adversarial Network for image generation in distributed scenes, Information Fusion, Volume 112,2024,102556,ISSN 1566-2535,https://doi.org/10.1016/j.inffus.2024.102556.