

Realistic Face Generation using a Textual Description

A MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF DEGREE
OF

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted by:

Anukriti (Roll No: 2K17/IT/027)
Anurag Mudgil (Roll No: 2K17/IT/029)
Nakul Dodeja (Roll No: 2K17/IT/074)

Under the supervision of

Dr. DINESH KUMAR VISHWAKARMA

Professor



DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

JUNE, 2021

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)


Bawana Road, Delhi-110042


CANDIDATE'S DECLARATION


We, Anukriti (2K17/IT/027), Anurag Mudgil (2K17/IT/029) and Nakul Dodeja (2K17/IT/074), students of B.Tech. (Information Technology), hereby declare that the project Dissertation titled “Realistic Face Generation using a Textual Description” which is submitted by us to the Department of Information Technology, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Date: 5th June, 2021

Place: New Delhi

Anukriti (2K17/IT/027) 

Anurag Mudgil (2K17/IT/029) 

Nakul Dodeja (2K17/IT/074) 

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled “Realistic Face Generation using a Textual Description” which is submitted by Anukriti (2K17/IT/027), Anurag Mudgil (2K17/IT/029) and Nakul Dodeja (2K17/IT/074) [Information Technology], Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.



Date: 5th June, 2021

Place: New Delhi

Dr. Dinesh Kumar Vishwakarma

SUPERVISOR

(Professor,
Department of Information Technology)

ABSTRACT

A renewed interest regarding the intersection of vision and language based tasks has been witnessed in recent years and amongst these areas, there is still a lot to do in the field of generating images from textual data. Although there are some existing works which focus in this domain, their primary focus is still birds or flower generation instead of human faces. This problem has a wide range of applications including movie/art creation, criminal face reconstruction, novel character transformation, photo-editing and many more.

In our work, we have successfully synthesized 128×128 resolution images from textual descriptions using a combined architecture of ProGAN and StackGAN. We incorporated both the architectures considering their strengths as StackGAN is known for its textual encoding and Progressive GAN is known for enhancing stability, variation as well as quality of images by incorporating layer by layer training, thus fine-tuning the details with the addition of an additional layer during the transition phase. We also created our customized dataset using CelebA for textual descriptions including facial features as well as expressions, incorporating diversity.

Furthermore, after 10 cycles of evaluation, we were able to get an inception score of 6.86 ± 0.06 and have shown promising results. The novel deep architecture we built in our work can be a significant addition towards bridging the breakthroughs in text and image modeling by generation of images using just textual descriptions while also maintaining stability.

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I am very thankful to Dr. Dinesh Kumar Vishwakarma (Associate Professor, IT) and all the faculty members of the Department of Information Technology of DTU. They all provided us with immense support and guidance for the project.

I would also like to express my gratitude to the University for providing us with the laboratories, infrastructure, testing facilities and environment which allowed us to work without any obstructions.

I would also like to appreciate the support provided to us by our lab assistants, seniors and our peer group who aided us with all the knowledge they had regarding various topics.

Date: 5th June, 2021



Place: New Delhi

ANUKRITI (2K17/IT/027)



ANURAG MUDGIL (2K17/IT/029)



NAKUL DODEJA (2K17/IT/074)

CONTENTS

CANDIDATE’S DECLARATION	ii
CERTIFICATE	iii
ABSTRACT	iv
ACKNOWLEDGEMENT	v
CONTENTS	vi
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF ABBREVIATIONS	x
CHAPTER 1: INTRODUCTION	1
1.1 Overview	1
1.2 Problem Statement	3
CHAPTER 2: LITERATURE REVIEW	5
CHAPTER 3: BACKGROUND	8
3.1 Discriminative Algorithms	8
3.2 Generative Algorithms	8
3.3 Generative Adversarial Networks	9
3.3.1 Architecture of GAN	9
3.4 StackGAN	12
3.5 Progressive GAN	14
3.6 Embedding	15

3.6.1 Word Embedding	15
3.6.2 Sentence Embedding	16
3.6.2.1 Sentence BERT	17
4. PROPOSED METHODOLOGY	18
4.1 Dataset and Structure of Data	18
4.2 Network Architecture	20
4.2.1 Textual Embedding Encoder Network	21
4.2.2 Conditioning Augmentation Block	22
4.2.3 Progressive Growing of GAN	23
4.2.3.1 Progressive Growing Discriminator Network	24
4.2.3.2 Progressive Growing Generator Network	26
4.2.3.3 Composite Model	28
4.3 Loss Functions	31
4.3.1 Wasserstein Loss	31
4.3.2 KL Loss Function	32
CHAPTER 5: RESULTS AND EXPERIMENTATION	33
5.1 Inception Score	33
5.2 Lost Variation	34
5.3 Visualization of Results	35
CHAPTER 6: CONCLUSION AND FUTURE WORK	38
REFERENCES	39
APPENDIX A: PUBLICATION DETAILS	43

List of Figures

Figure 3.1: Architecture of GAN.....	10
Figure 3.2: Architecture of StackGAN.....	13
Figure 3.3: Architecture of ProGAN.....	14
Figure 3.4: Word embedding Visualization.....	15
Figure 3.5: Siamese neural network in Sentence BERT.....	17
Figure 4.1: Proposed Architecture.....	21
Figure 4.2: Transition from 16x16 images to 32x32 images.....	30
Figure 5.1: Variation of training loss with the number of training epochs.....	35
Figure 5.2: Visual results of architecture layer by layer.....	36
Figure 5.3: Visual results with textual descriptions.....	37
Figure A.1: Proof of publication.....	43

List of Tables

Table 4.1: Description of dataset.....	19
Table 4.2: Description of shape of embeddings.....	22
Table 4.3: Discriminator Architecture.	25
Table 4.4: Generator Architecture.....	27
Table 4.5: Hyperparameters.....	29

List of Abbreviations

Abbreviation	Explanation
BERT	Bidirectional Encoder Representation from Transformers
CDR-GAN	Chained Deep Recurrent Generative Adversarial Network
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
DAMSM	Deep Attentional Multimodal Similarity Model
DCGAN	Deep Convolutional Generative Adversarial Network
DCNN	Deep Convolutional Neural Network
DR-GAN	Disentangled Representation Learning Generative Adversarial Network
GAN	Generative Adversarial Network
GB	Giga Bytes
HDGAN	High Definition Generative Adversarial Network
IS	Inception Score
KL divergence	Kullback-Leibler divergence
LD-CGAN	Lightweight Dynamic Conditional GAN
LFW	Labelled Faces in the Wild
RNN	Recurrent Neural Network
SBERT	Sentence Bidirectional Encoder Representation from Transformers
SD	Standard Deviation
VAE	Variational Auto Encoder
WGAN-GP	Wasserstein GAN - Gradient Penalty

CHAPTER 1

INTRODUCTION

The following chapter provides an introduction to the conducted research. Section 1.1 provides an overview of the problem domain and Section 1.2 presents our problem statement.

1.1 OVERVIEW

The intellect of humans enables them to imagine an object based on their description but the image generated in our mind is usually quite vague and lacks important details. With the rise in the field of machine learning it has made the machines so “intelligent” that now they have become capable of generating images just on the basis of their description. The major architecture that solves this field of problems are Generative Adversarial Networks which apart from this many other applications like generating deep fake videos , transforming images between two different animals, image translation etc.

Generation of images just on the basis of their textual description has many applications in our day to day lives.

We have found many texts from historical times but most of the portraits of the emperors or significant of that time are either destroyed or were never created. The pictures that we see now are designed based on the descriptions that we have found from the text but they are still not that accurate. Text to image generation helps us to generate more realistic images of the people in the past based on the descriptions we find in historical texts.

Text to image generation is complete opposite of another machine learning application i.e. Image captioning where given an image we can get the important details from it in the form of text. Image captioning in combination with text to image generation has many applications in the field of image or video editing like if you have an image and you want some of the details to be changed then first you can generate the text from the original image using Image captioning, make the required changes in the text and then generate the image again this time from this changed text.

Sketch artists are people that can draw the portrait of the person based on the descriptions explained to them. They have their uses usually in the police stations where they draw the sketches of criminals or lost people. No matter how accurate the sketch artist is, it often fails to draw a realistic image of the person and just gives a vague idea of their description. Text to image conversion using AI has a significant application in this field.

When movies are made based on a novel , casting is one of the most difficult tasks and the director has to make sure that the actors in the movies match the characters in the book based on the physical descriptions provided in the book. Generating the image from the description assists the director to choose a more realistic character for the role.

Apart from all these application text to image generation techniques has immense usage in the field of computer aided design.

In this work we focus primarily on the generation of human faces based on their textual descriptions. Already some work has been done in text to image generation but they primarily focussed on generating the pictures of animals or birds or some scene. Human face is a much more complex image compared to these with a large number of small details that can easily be missed. These features may include color of

eye, shape of nose, hair style, complexion etc. So the field of generation of human faces is a relatively new one and hasn't been explored by many.

1.2 PROBLEM STATEMENT

Most of the initial algorithms of machine learning focused on supervised learning where we can get insights from a dataset which focuses on finding a relation between the features and the labels of the data set. Image captioning , image classification are few of the common examples that work on this principle and they have many applications in our lives like Google photos ,virtual assistance etc.

The advancement to this principle and a more complex problem can be its inverse. That is generation of data based on the insights of the data. This focuses on generating a data point which is never seen before just on the basis of its description. Text to image conversion is one of the important examples of such problems.

With the introduction of many generative models many models have been developed that solves the problem of generating the image from its description. The problem with initial models was that they generated a vague image that just provided the basic meaning of their description and often couldn't show the details and vivid object parts. These models usually worked on generation of images of birds or animals where the images are less detailed and they produced satisfactory results. But in the cases where even small details are of significant importance like generation of faces based on description these models failed to perform well. With an increase in resolution further, the problem become more severe and starts giving nonsensical outputs.

The primary intention of this research is Text to Face Generation - generating a X b resolution image of faces of people on the basis of the textual description provided.

We have transformed the basic features of faces like hair color, face color etc. as well as more advanced features like mature face , shape of nose, arched eyebrows into an image of an actual person which has these features. Not only limiting this the expressions on the faces of the person like a happy face with a wide smile, sad or tense face etc. as described in the description is also depicted in its corresponding image.

CHAPTER 2

LITERATURE REVIEW

The following chapter presents a literature review corresponding to the domain of our problem statement.

Learning from multimodal data involves conditioning and prediction of features over multiple domains like audio, image, and text [1], [2]. Text to Image Generation and Text to Face Synthesis are examples of the same.

Autoregressive Convolutional Neural Networks (CNN), Variational AutoEncoders (VAE), and Generative Adversarial Networks [3] have been traditionally used as generative mechanisms. CNNs, like PixelCNN [4], slowly train conditional distribution over pixels in a direct manner, restricting their relevance. Variational AutoEncoders [5] tend to produce images of low resolution but are easy to train. Generative Adversarial Networks (GANs) [3] undergo a less stable training process but generate sharp images. Even hybrid models like PixelGAN autoencoders [6] and CVAE-GAN [7] that combine advantages of more than of the three mechanisms fail to produce images better in quality than GANs. Different variants of Generative Adversarial Networks (GAN) and Conditional Generative Adversarial Networks (cGANS) [8] have consequently been developed.

The nascent work [9] done in this field involved training of a deep neural model akin to DCGAN [10] that is conditioned on text encoded using a character-level CRNN. It was able to convert natural language into 64×64 images. Some of the authors of this work proposed an advancement [11] that was able to produce 128×128 images from constraints like object location and textual description. Work done by Reed et al. has inspired others to generate images using the combination of disentangled semantics of input image and text modalities [12], intermediate inferred semantic layout [13] and dialogue [14].

A major development came with StackGAN [15] that was able to produce realistic 256×256 images by dividing the synthesis process into two levels. An improvement known as StackGAN-v2 [16] allows prediction over more than one distribution for both conditional and unconditional image synthesis. AttnGAN [17] utilizes the mechanism of attention for rendering images in accordance with the keywords but with longer input text, the poor training of attention map leads to faltering efficiency. MirrorGAN [18] employs re-description for semantic preservation via a model collaborating global and local attention [19]. HDGAN proposes the usage of a deep hierarchically-nested network of discriminators to generate images of 512×512 resolution. LD-CGAN [20] generates 128×128 images with comparable evaluation performance and decreased parameters and computation time as compared to HDGAN. CDRGAN [21] makes use of cascaded recurrent generator network and convolutional discriminator network for better semantic inference and reduced computational resources. Recently, models involving unsupervised [22] and contrastive learning [23] for text-to-image synthesis have also been developed.

Several works involve the synthesis of facial images from low-resolution images and/or attribute vectors. DCGAN [10] suggests constraints on the architecture of conditional GAN and assesses them on faces dataset for better training on unlabeled data. ProGAN [24] facilitates faster, stable training and continuous improvement in the details and resolution of faces by adding extra layers as the training progresses. However, it doesn't come close to photorealism and understanding dataset-specific conditions. StyleGAN [25] presents a generator inspired by the transfer of style for usage in GANs that possesses better disentanglement between stochastic (like, hair) and high-level descriptive attributes (like, pose). StarGAN [26] employs a single architecture for performing image-to-image translation over multiple domains. Conditional CycleGAN [27] generates high-resolution facial images from the input of low-resolution facial images and vector of high-resolution attributes.

The aforementioned models can generate facial images of decent quality but they do not perform synthesis from a textual description of facial features. Unlike flowers [28] and birds [29] that have a finite variety and easy descriptors like color, mapping text to a face is a much more complex task due to inherent diversity in features like ethnicity, age, and

expression. Though, there are several datasets for text to image generation [30], the lack of substantially large and annotated datasets related to faces acts as another reason behind limited work in the field of text to face generation.

Fully Trained GAN (FTGAN) [31] generator performs text encoding using BiLSTM and image decoding using 3-stage CNN simultaneously and has 3 discriminators for generating images of varying resolutions. DAMSM network used in AttnGAN [17] is also used here for text encoding and calculation of attention map, thereby leading to the problem of unstable training, especially for longer text. Another fully trained GAN proposed by Khan et al. [32] has a lower Fretchet Inception Distance score than FTGAN indicating better images but doesn't perform evaluation over important Inception Score metric and has potential for incorporating dense attribute details. TTF-HD [33] is a framework that includes a text classifier of multiple annotations, an encoder of image annotations, and an image generator StyleGAN2 [34] capable of feature disentanglement for the generation of a set of high-resolution diverse facial images. However, due to a lack of training data, it suffers from a low inception score, insufficient semantic consistency, and inaccuracies in text classification, image encoding, and feature disentanglement.

CHAPTER 3

BACKGROUND

The following chapter provides the background to the conducted research. Section 3.1 discusses discriminative algorithms. Section 3.2 discusses generative algorithms. Section 3.3 discusses generative adversarial networks and their architecture. Section 3.4 discusses Stack GAN and its architecture. Section 3.5 discusses Progressive GAN and its architecture. Section 3.6 discusses word and sentence embeddings along with Sentence BERT architecture.

3.1 DISCRIMINATIVE ALGORITHMS

There has been lots of research on discriminative algorithms in recent times. They are the algorithms that basically focus on distinguishing or classifying data instances. More precisely, they help find the labels of data on the basis of their features. Discriminative focuses on finding the conditional probability $p(Y | X)$.

For example, On the basis of a photograph deciding whether it is a cat or a dog.

3.2 GENERATIVE ALGORITHMS

The reverse of discriminative models is known as generative models. That means on the basis of the labels we have to find the feature set for the given label. They include distribution of data itself and generate the data from random noise or uniform distribution. The main focus of generative models is to calculate Joint Probability $p(X, Y)$. In case labels are absent, they aim to find $p(X)$.

For example, On the basis of whether a cat or dog was input, generating a relevant picture of the animal.

3.3 GENERATIVE ADVERSARIAL NETWORKS

General Adversarial Networks (abbreviated as GAN) are most ordinarily known examples of generative models that were introduced by Ian Goodfellow and his colleagues in the year 2014. According to Yann Le Cun (Director of AI at Facebook), GANs are representative of the most riveting innovations in the field of machine learning in the past decade.

3.3.1 Architecture of GAN

The main idea behind GAN is two neural networks contesting with each other. These two neural networks are called generator (the artist) and discriminator (the art critic). Generator is a neural-network model that aims to synthesize images that are fake in nature by means of random noise or uniform data. Then these fake images by generator and training images act as input to the Discriminator which is a traditional neural network whose main aim is to discern between the real and fake images. Thus the primary goal of the Generator is to fool the discriminator and that of the discriminator is to prevent itself from getting fooled. The discriminator then provides feedback to the generator to make images more close to the real ones. When training begins the generator forms obviously fake instances and discriminators easily distinguish between them but as the training continues both generator and discriminator becomes stronger and stronger but at some point of time generator becomes stronger than discriminator based on the feedback provided by the discriminator and starts generating images so close to that of real images that demarcation between images of fake nature from those that are real becomes an

unattainable task. It is the time when $p(X|Y)$ for discriminator becomes 0.5 (that is it randomly classifies images as real and fake). It is important to stop the training at that point because then the feedback from the discriminator is of no use and it may lead to a deterioration of our model.

Fig. 3.1 represents the architecture of GAN.

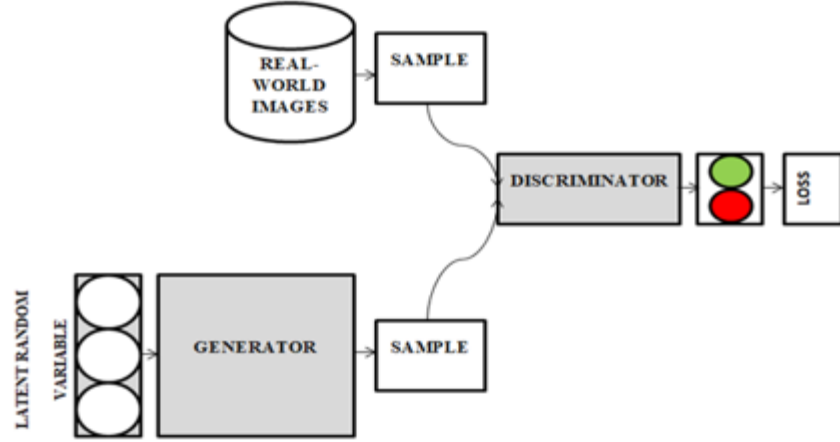


Figure 3.1: Architecture of GAN

This process is called Adversarial Training.

<p style="text-align: center;">At Discriminator D</p> $\text{Disc_loss}_{\text{real}} = \log(\text{Disc}(x))$ $\text{Disc_loss}_{\text{fake}} = \log(1 - \text{Disc}(\text{Gen}(\text{Gen}(z))))$ $\text{Disc_loss} = \text{Disc_loss}_{\text{real}} + \text{Disc_loss}_{\text{fake}}$ $= \log(\text{Disc}(x)) + \log(1 - \text{Disc}(\text{Gen}(\text{Gen}(z))))$ <p style="text-align: center;">The total cost is</p> $\frac{1}{n} \sum_{i=1}^n \log(x^i) + \log(1 - \text{Disc}(\text{Gen}(z)))$	<p style="text-align: center;">At Generator G</p> $\text{Gloss} = \log(1 - \text{Disc}(\text{Gen}(z))) \text{ or } -\log(\text{Disc}(\text{Gen}(z)))$ <p style="text-align: center;">The total cost is</p> $\frac{1}{n} \sum_{i=1}^n \log(1 - \text{Disc}(G(z^i)))$
--	---

(3.1)

$$\min_G \max_D F(\text{Disc}, \text{Gen}) = E_{x \sim p_{\text{data}}(x)} [\log \text{Disc}(x)] + E_{x \sim p_z(z)} [\log(1 - \text{Disc}(\text{Gen}(z)))]$$

(3.2)

$$\begin{aligned}
\max_D F(\text{Disc}) &= E_{x \sim p_{\text{data}}(x)} [\log \text{Disc}(x)] + E_{x \sim p_z(z)} [\log(1 - \text{Disc}(\text{Gen}(z)))] \\
\min_G F(\text{Gen}) &= E_{x \sim p_z(z)} [\log(1 - \text{Disc}(\text{Gen}(z)))]
\end{aligned}
\tag{3.3}$$

Here, the equation 3.1 represents the loss function for Discriminator where $D(x)$ is maximized whereas $D(G(x))$ is minimized. It also represents the actual cost function which will be minimized while training. Similar to the discriminator, at generator also, it represents generator loss but contrary to discriminator, here $D(G(z))$ is maximized.

So in terms of game theory, the discriminator and generator play the minimax game where the value function is $V(D,G)$ where the generator tries to minimise the value of the function given in Equation 3.3 while the discriminator tries to maximize this value as shown. The corresponding minimax equation is given by Equation 3.2.

So, in terms of game theory, the discriminator and generator play the minimax game where the value function is $V(D,G)$ where the generator tries to minimise the value of the function while the discriminator tries to maximize this value.

There are various types of GAN based on their architecture:

- Deep Convolutional GANs (DCGANs)
- Cycle GAN
- Vanilla GAN
- Conditional GAN (C-GAN)
- Discover Cross-Domain Relations with Generative Adversarial Networks (Disco GANS)

3.4. STACKGAN ARCHITECTURE

Stack GAN is one of the types of GANs which generates one of the most high quality images and is most frequently used in GAN architecture. The basic working of stack GAN can be described as breaking down a complex problem of generation of high resolution images into smaller manageable pieces and the slowly improving state of art.

This model comprises two components:

$$\begin{aligned} \text{Encoder } Y &= E(x) \text{ where } x \text{ is the image and } y \text{ is its label, and} \\ \text{A decoder } X &= G(y, z) \text{ where } z \text{ is the noise} \end{aligned} \tag{3.4}$$

The architecture is shown in Figure 3.2.

Stage 1 GAN: It is the first stage where we generate and find the basic shapes and basic colors of the image based on the textual description provided to it. Similarly it also generates a low resolution background from the noise vector thus generating a low resolution image.

It trains by alternatively reducing the loss functions for the generator and maximising loss function of the discriminator.

$$L_{\text{Disc}} = E_{(I, t) \sim p_{\text{data}}} [\log \text{Disc}(I, \phi_t)] + E_{s_0 \sim p_{\text{Gen}_0}, t \sim p_{\text{data}}} [\log(1 - \text{Disc}(\text{Gen}(s_0, \hat{P}), \phi_t))] \tag{3.5}$$

$$L_{Gen} = E_{s_0 \sim p_{G_0}, t \sim p_{data}} \left[\log \left(1 - \text{Disc}(\text{Gen}(s_0, \hat{P}), \phi_t) \right) \right] + \lambda \text{Disc}_{KL} \left(N(\chi(\phi_t), \sum (\phi_t)) \parallel N(0, I) \right) \quad (3.6)$$

$$L_{\text{Disc}_0} = E_{(I_0, t) \sim p_{data}} \left[\log \text{Disc}_0(I_0, \phi_t) \right] + E_{x \sim p_x, t \sim p_{data}} \left[\log \left(1 - \text{Disc}_0(\text{Gen}_0(x, \hat{P}_0), \phi_t) \right) \right] \quad (3.7)$$

$$L_{\text{Gen}_0} = E_{x \sim p_x, t \sim p_{data}} \left[\log \left(1 - \text{Disc}_0(\text{Gen}_0(x, \hat{P}_0), \phi_t) \right) \right] + \lambda D_{KL} \left(N(\chi_0(\phi_t), \sum_0(\phi_t)) \parallel N(0, I) \right), \quad (3.8)$$

Equation 3.5 represents the loss function for the discriminator network, in which both networks are conditioned on the text embeddings. One major difference is that the generator network has x and c as inputs, where x is the image generated by the Stage-I and c is the Conditioning Augmentation variable.

Equation 3.6 here represents the loss function for the generator network in Stage-II GAN. This network also consists of a Kullback-Leibler (KL) divergence term to its loss function.

Equation 3.7 represents the loss function for the discriminator network, in which both networks are conditioned on the text embeddings.

Equation 3.8 represents the loss function for the generator network in Stage-I GAN, in which both networks are conditioned on the text embeddings. Also, it includes a KL divergence term to the loss function.

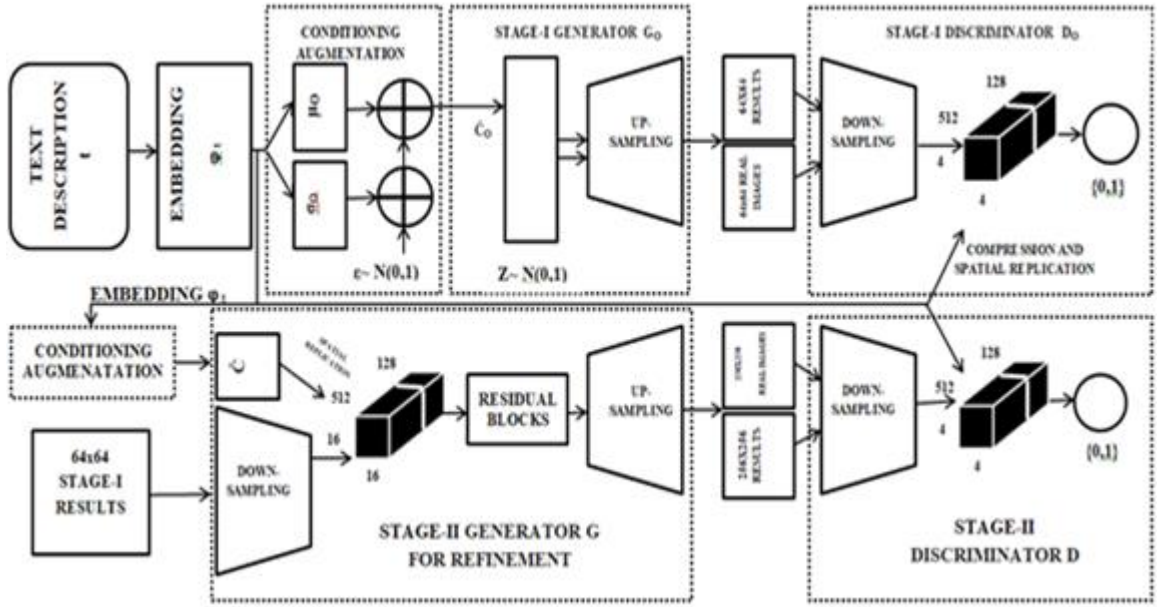


Figure 3.2: Architecture of StackGAN

3.5 PROGRESSIVE GAN ARCHITECTURE

Progressive Growing GAN is an extension of the GAN training procedure which trains generator models with stability and also yields large high resolution images. It involves starting off with a very small image and gradually adding layers blocks so that the output size of the generator model grows and the input size of the discriminator model grows until the required image size is reached. This method has shown to be very effective in producing high-quality synthetic images that are extremely realistic.

The Progressive Growing GAN's main novelty is the gradual rise in the size of images produced by the generator, starting with a 4X4 pixel picture and then doubling until the desired output resolution is reached. This is accomplished by a training approach that alternates between fine-tuning the model with a certain output resolution and gradually phasing in a new model with a higher resolution. During the

training phase, it adds additional convolutional layer blocks to both the generator and discriminator models in a methodical manner. On both the generator and discriminator sides, this gradual addition of convolutional layers allows the models to learn coarse-level detail effectively at first and then learn increasingly finer detail afterwards. All levels, including previous layers, remain trainable, even if new layers are introduced.

This architecture is demonstrated in Figure 3.3.

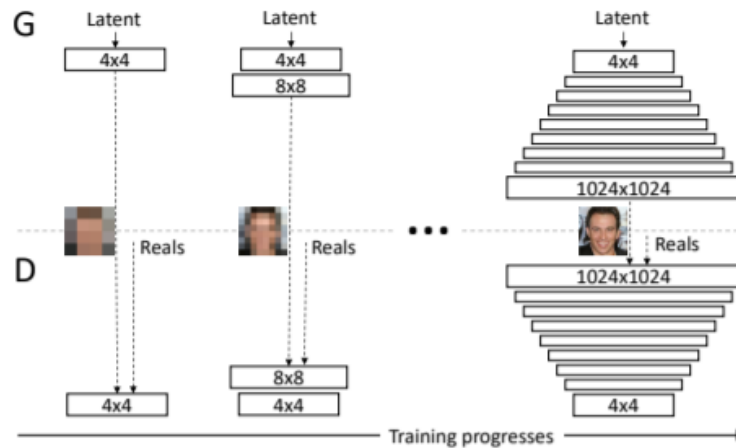


Figure 3.3: Architecture of Progressive GAN

3.6 EMBEDDING

3.6.1 Word Embedding

We can encode the text by generating a sparse matrix for the whole vocabulary where each entry corresponds to a word and we can represent a word by an array with all 0s and 1 at a unique position describing the word. But this thing which works well for digits is not suitable with words because this method does not

distinguish between related and unrelated words therefore we use a different method for words which also represent a word meaning a numeric vector but now two similar words have similar corresponding vectors. It is possible to do analogies like king is to man is same as queen is to woman. There are different methods for word embedding like Word2vec, GloVe, and FastText etc. Fig. 3.4 represents a visualization of word embeddings.

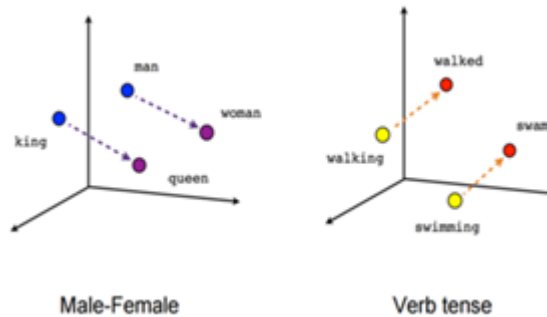


Figure 3.4: Word embedding Visualization

3.6.2 Sentence Embedding

Many times we need to take a step ahead and cypher the meaning of the whole sentence. This is known as sentence embedding. It allows us to not calculate individual word embedding but rather see sentences as a whole. It is a very widely used technique for sentiment analysis or predicting the values. Calculating centroid distance was once a widely used technique for sentence embedding but its limitation to distinguish between the sequence of two sentences has made it very less popular.

Sentence-BERT is one of the most widely used sentence embedding techniques nowadays.

3.6.2.1 Sentence BERT

Sentence BERT is very good at making embeddings because of its adaptability. The word embeddings change dynamically on the basis of the vector surrounding them, thus two sentences I ate an apple and Apple bought a startup have completely different embeddings because of the different context. It gets trained using a sentence with added noise where it aims to bring back the sentence into its initial form.

This training is done on GBs of data from various sources. To give good results fine tuning is applied on BERT which means that we reward the model with sentence embedding having the property that cosine similarity for pairs of sentences represents semantic similarity for the sentences. If we find such a model then we can generate sentence embedding once and then find the cosine similarity for each pair. This can be done with the help of Siamese Neural Network where we obtain embedding for pairs of sentences and then concatenate them. Fig. 3.5 represents SBERT trained for classification objectives.

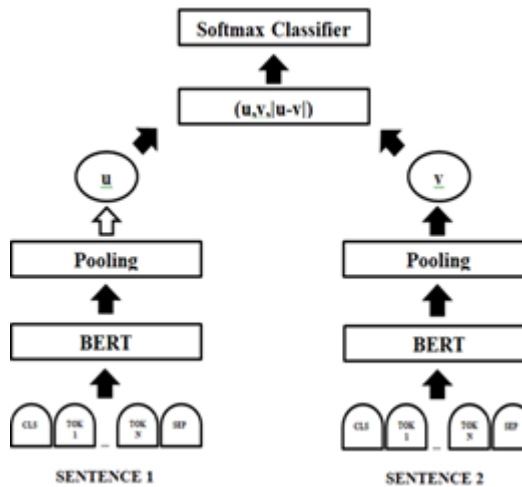


Figure 3.5: Siamese neural network in Sentence BERT

CHAPTER 4

PROPOSED METHODOLOGY

The following chapter aims to explain the work done in detail. It explains how a combined architecture of ProGAN and StackGAN has been used to generate highly realistic images from textual description. Section 4.1 explains how relevant data for our problem domain has been created. Section 4.2 represents our model architecture and its major components. This also contains implementation details of our architecture, discussing the loss functions associated as well as their corresponding optimizers.

4.1 DATASET AND STRUCTURE OF DATA

The first setback that we observed in our study was the unavailability of image-description datasets. We still have some image captioning datasets for scene or object descriptions such as MS COCO [35], Flickr30k [36] and VLT2K [37]. Similarly, we also have datasets for visual representations of flowers and birds such as the most commonly used Oxford Flowers- 102 and Caltech-UCSD Birds but none of these are of any use in our work as they do not specifically focus on face descriptions.

In order to obtain a dataset capturing fine-grained facial details which could rely on attributes rather than just the object and relations, we used one of the popular datasets - CelebFaces Attributes Dataset (CelebA). We decided to use this dataset as our base for generating textual descriptions due to it being a large-scale dataset which could help us improve the process of training and testing.

It contains the facial features of around 2,02,599 celebrity images, each with a wide range of posture variations and diversity. These images contain 40 features per image

and cover approximately 10,177 identities as well as five landmark positions. This dataset is frequently used as training and testing subsets in a variety of computer vision applications.

In our study, we divided the 40 attributes list in the CelebA dataset corresponding to each image into six groups, each of which described a unique face feature. The facial shape, haircut, and appearance descriptive qualities were used to construct these categories (such as young, pale etc.) accessories worn, the person's hairstyle, and other facial attributes including eyes, lips, cheeks and so on. The gender attribute is used to identify a person as male or female and to indicate whether to use the pronouns 'he' or 'she' in textual descriptions. A dictionary is also constructed, with keys representing CelebA dataset attributes and values representing the text with which we want to replace them in the descriptions.

While constructing a description, we first included a prefix that corresponded to the unique group (For example, "He has a " to describe the oval face attribute) and then the relevant value from the dictionary was added to it. We ultimately received the whole textual description highlighting the person's facial features, face outline, and relevant 13 accessories based on which we generated photos after summing up all of these individual statements about an individual. The dataset that was used is described in Table 4.1.

Table 4.1: Description of dataset

Number of pictures of celebrities' faces	202599
Unique Images	10177
Binary Attribute Labels per Picture	40
Significant Locations	5

4.2 NETWORK ARCHITECTURE

Our work began with the two-staged StackGAN architecture [15] as it is known for its textual encoding network. However, it came up with a few limitations including failure convergence and mode collapse problems. In order to handle these, we decided to use the progressive growing technique of Progressive GAN [24] for our model architecture. This was one step closer towards reaching our goal of quality and stability while training GANs. Hence, in order to achieve the best of both worlds, we came up with an architecture combining the advantages of both the StackGAN as well as Progressive GAN (ProGAN).

The major components of StackGAN that we incorporated are textual encoder network, embedding compressor and Conditioning Augmentation block. We further used Progressive GAN for allowing the generator models to train with stability which can produce large-high-quality images by increasing spatial resolutions layer by layer.

The subsequent sections explain each of the components of our proposed architecture in detail. Using them, we created fake but photo-realistic looking images after training on image data as they are used to learn a conditional density model. Figure 4.1 represents the architecture of the proposed model.

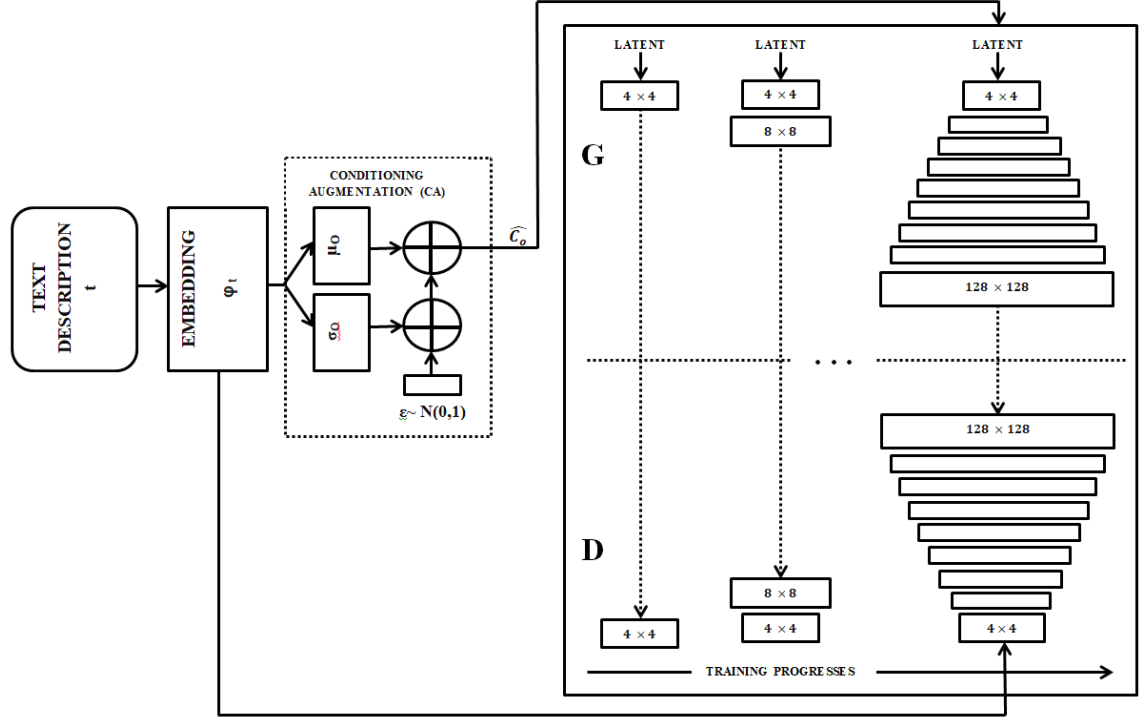


Figure 4.1: Proposed Architecture

4.2.1 Textual embedding encoder network

The initial stage, as illustrated in Fig. 4.1, was to encode the text description into an embedding, for which we utilized a Sentence-BERT network that had already been trained. Figure 3.4 In order to create a fixed size sentence embedding, it adds pooling to the token embeddings provided by BERT. These can be used to encode the semantics of sentence data in unsupervised tasks. The main goal of embedding is to make it easier for people to find information.

The primary goal of embeddings is to generate constant-size vectors from random-size input text. A sentence is encoded into a 1,024 dimensional text embedding via the text encoder network. This network is present in both stages. The shape of the training and test set embeddings is shown in Table 4.2.

Table 4.2: Description of shape of embeddings

Train_emb_shape	(151949,10,1024)
Test_emb_shape	(50650,10,1024)

4.2.2 Conditioning augmentation block

Text-conditioned latent space often has a large number of dimensions, which can lead to discontinuities. To make the production of supplementary variables for conditioning easier, we used the Conditioning Augmentation approach to extract latent variables from text embeddings. The conditioning augmentation block in this methodology represents a single linear layer into which embeddings of shape (1024,) are supplied and a tensor of shape (256,) is returned, which is considered the relevant input to be processed by GAN. This network is used to sample random latent variables from the following distribution:

$$\mathcal{L}(\chi_0(\phi_t), \sum_0(\phi_t)) \quad (4.1)$$

This strategy increases the network's unpredictability. Also contributes to the generator's robustness by capturing items with a wider range of positions and looks. We can train a resilient network that can handle perturbations with a larger number of image-text pairs. We examined not only the training photos and noise inspiration, but also some additional information denoted by c , which in our case was the term used to characterize each image. The discriminator uses this information to see if the phrase matches the image, and the generator uses it as inspiration for producing the image.

Following the acquisition of textual embeddings from Sentence-Bert, a fully-connected layer is used to create values such as the mean and standard deviation. These are then placed in a diagonal of the matrix to generate a diagonal covariance matrix. Finally, we use these to form a Gaussian distribution, which looks like this:

$$\mathcal{L}(\chi_0(\phi_t), \sum_0(\phi_t)) \quad (4.2)$$

After that, we took a random sample from the Gaussian distribution we had just produced. To sample \hat{P}_0 , we multiply the standard deviation element-by-element and then add the result to the mean. To calculate, we used the following formula:

$$\hat{P}_0 = \chi_0 + \sigma_0 \odot N(0, I) \quad (4.3)$$

4.2.3 Progressive Growing of GAN

A GAN is typically made up of two networks: a generator and a discriminator. The generator generates a sample, such as an image, from a latent coding, and the discriminator network is trained to see if the distribution of these images is indistinguishable from the training distribution. However, there are a number of issues with this formulation.

Firstly, high-resolution image generation is problematic since higher resolution makes it easier to distinguish generated images from training images, aggravating the gradient problem. Furthermore, we find that mode collapses, which have long been a problem for GANs, occur quite quickly, over the course of a dozen mini batches. They usually begin when the discriminator overshoots, resulting in

exaggerated gradients and an unhealthy rivalry in which the signal magnitudes in both networks increase.

We decided to use Progressive GANs in order to overcome these challenges during the training phase, wherein we start with low-quality images and gradually increase the quality by adding layers to the networks. Instead of learning all scales at once, this incremental approach allows the training to first uncover the image distribution's large-scale structure and then transfer attention to increasingly finer scale information. In our study, we employed generator and discriminator networks as mirror images of each other with both growing in sync. Throughout the training process, all existing layers in both networks were trainable. When new layers were introduced to the networks, they gradually faded in.

4.2.3.1 Progressive Growing Discriminator Network

The discriminator model used in our work is a deep convolutional neural network that takes as an input a 4×4 -colored image and predicts whether it is real or not. A 1×1 convolutional layer was the initial hidden layer. In the output block, we have used a MinibatchStdev used for summarizing the batch statistics for the batch of images provided in the discriminator. Also, 3×3 and 4×4 convolutional layers along with a fully connected layer were used to output the final prediction. After all layers, leaky ReLU activation functions were utilised, while the output layers employ a linear activation function.

This model was trained further for intervals after which the model went through a phase upto 8×8 which entailed the addition of two blocks of 3×3 convolutional layers along with an average pooling downsample layer. The input image passed through the new block with a new 1×1 convolutional hidden layer. It was also passed through a downsample layer and through the old 1×1 convolutional

hidden layer. A WeightedSum layer was used to control the weighted sum of the previous 1×1 convolution layer and new layers during a growth phase.

The initialization of all layers was done with small Gaussian random numbers, with a standard deviation of 0.02 and the value of maxnorm weight constraint set as 1. We have also defined a number of filters which increase as the depth of the model goes on increasing from 16 to 32, 64 and so on. Each model was compiled and fitted using stochastic gradient descent and Wasserstein loss about which we will be discussing in Section 4.3.1.

Table 4.3: Discriminator Architecture

Discriminator	Activation Function	Output Shape
Input_Image	--	3 X 128 X 128
Conv_1 (1 X 1)	LReLU	4 X 128 X 128
Conv_1 (3 X 3)	LReLU	4 X 128 X 128
Conv_1 (3 X 3)	LReLU	8 X 128 X 128
Downsampling_Layer1	--	8 X 64 X 64
Conv_2 (3 X 3)	LReLU	8 X 64 X 64
Conv_2 (3 X 3)	LReLU	16 X 64 X 64
Downsampling_Layer2	--	16 X 32 X 32
Conv_3 (3 X 3)	LReLU	16 X 32 X 32
Conv_3 (3 X 3)	LReLU	32 X 32 X 32
Downsampling_Layer3	--	32 X 16 X 16
Conv_4 (3 X 3)	LReLU	32 X 16 X 16
Conv_4 (3 X 3)	LReLU	64 X 16 X 16
Downsampling_Layer4	--	64 X 8 X 8

Conv_5 (3 X 3)	LReLU	64 X 8 X 8
Conv_5 (3 X 3)	LReLU	64 X 8 X 8
Downsampling_Layer5	--	64 X 4 X 4
Minibatch_Stdev	--	64 X 4 X 4
Conv_6 (3 X 3)	LReLU	64 X 4 X 4
Conv_6 (4 X 4)	LReLU	64 X 1 X 1
Fully_connected_Layer	Linear function	64 X 1 X 1

After a training interval, the alpha parameter (WeightedSum layer) transitioned from 0 to 1, another training phase was then run to adjust the new model without the old layer and pathway. This operation was repeated until the desired image size was reached, which in our case was 128×128 pixels.

The architecture of our discriminator network is described in Table 4.3.

4.2.3.2 Progressive Growing Generator Network

The generator network is defined in the same way as our discriminator network wherein it used a random point from the latent space as input and further, generated a synthetic image. A base model was defined initially for generating 4x4 images and then, for the larger image output size, growth versions were created.

The primary distinction between the two is that during the growth phase, the model's output is the same as the WeightedSum layer's output. The model's growth phase version begins with the addition of a nearest neighbour upsampling layer, which is then connected to the new block with the new output layer and the

old old output layer. A WeightedSum output layer is then used to combine the previous and new output layers.

A fully connected layer with a sufficient number of activations was defined as an input block in the base model to construct a specific number of 4×4 feature maps. Following that were 4×4 and 3×3 convolution layers, as well as a 1×1 output layer that provided us with colored images. New blocks were added with an upsampling and two 3×3 convolutional layers.

Table 4.4: Generator Architecture

Generator	Activation Function	Output Shape
Latent_vector	--	128 X 1 X 1
Conv_1 (4 X 4)	LReLU	128 X 4 X 4
Conv_1 (3 X 3)	LReLU	128 X 4 X 4
Upsampling_Layer1	--	128 X 8 X 8
Conv_2 (3 X 3)	LReLU	64 X 8 X 8
Conv_2 (3 X 3)	LReLU	64 X 8 X 8
Upsampling_Layer2	--	64 X 16 X 16
Conv_3 (3 X 3)	LReLU	32 X 16 X 16
Conv_3 (3 X 3)	LReLU	32 X 16 X 16
Upsampling_Layer3	--	32 X 32 X 32
Conv_4 (3 X 3)	LReLU	16 X 32 X 32
Conv_4 (3 X 3)	LReLU	16 X 32 X 32
Upsampling_Layer4	--	16 X 64 X 64
Conv_5 (3 X 3)	LReLU	8 X 64 X 64
Conv_5 (3 X 3)	LReLU	8 X 64 X 64

Upsampling_Layer5	--	8 X 128 X 128
Conv_6 (3 X 3)	LReLU	4 X 128 X 128
Conv_7 (3 X 3)	LReLU	4 X 128 X 128
Conv_8 (1 X 1)	Linear function	3 X 128 X 128

Also, the number of feature maps decreased with the depth of the model as opposed to our discriminator. However, weight initialization is the same as the ones in our discriminator with standard deviation of 0.02 and maxnorm weight constraint value of 1. Our activation function is the LeakyReLU, and the PixelNormalization layer is used after each convolutional layer. The output layer uses a linear activation function rather than the more usual tanh function, although real images are still scaled to the range $[-1,1]$, as is customary for most GAN models.

The architecture of our generator network is described in table 4.4.

4.2.3.3 Composite Model

Since the generator models are not trained directly, they could not be compiled. Instead, they were trained using Wasserstein loss via the discriminator models. This involved presenting the discriminator our images from the generator as real images and calculating the loss, which was then utilized to update the generator models. While doing so, we just needed to keep in mind that our generator and discriminator should be paired up in terms of the same image size such as 4x4 or, 8x8 as well as training phase - introducing the new block during growth phase or during fine-tuning phase. Table 4.5 indicates the hyperparameters used while training the model.

Table 4.5 Hyperparameters for training

Parameters	Value
caption_len	100
img_dims	[128,128]
CA_output_size	178
progan_depth	6

This new layer added in the growth phase utilizes the fade-in technique which helps to remember and restore previously learned information. Figure 4.2 indicates this technique during transition from 16x16 images to 32x32 images. Using nearest neighbour filtering and average pooling, the image resolution was doubled and halved respectively. While training the discriminator, we utilized real images that had been downscaled in order to match the network's current resolution. We interpolated between two resolutions of the real images during the transition, similar to how the generator's output combined two resolutions.

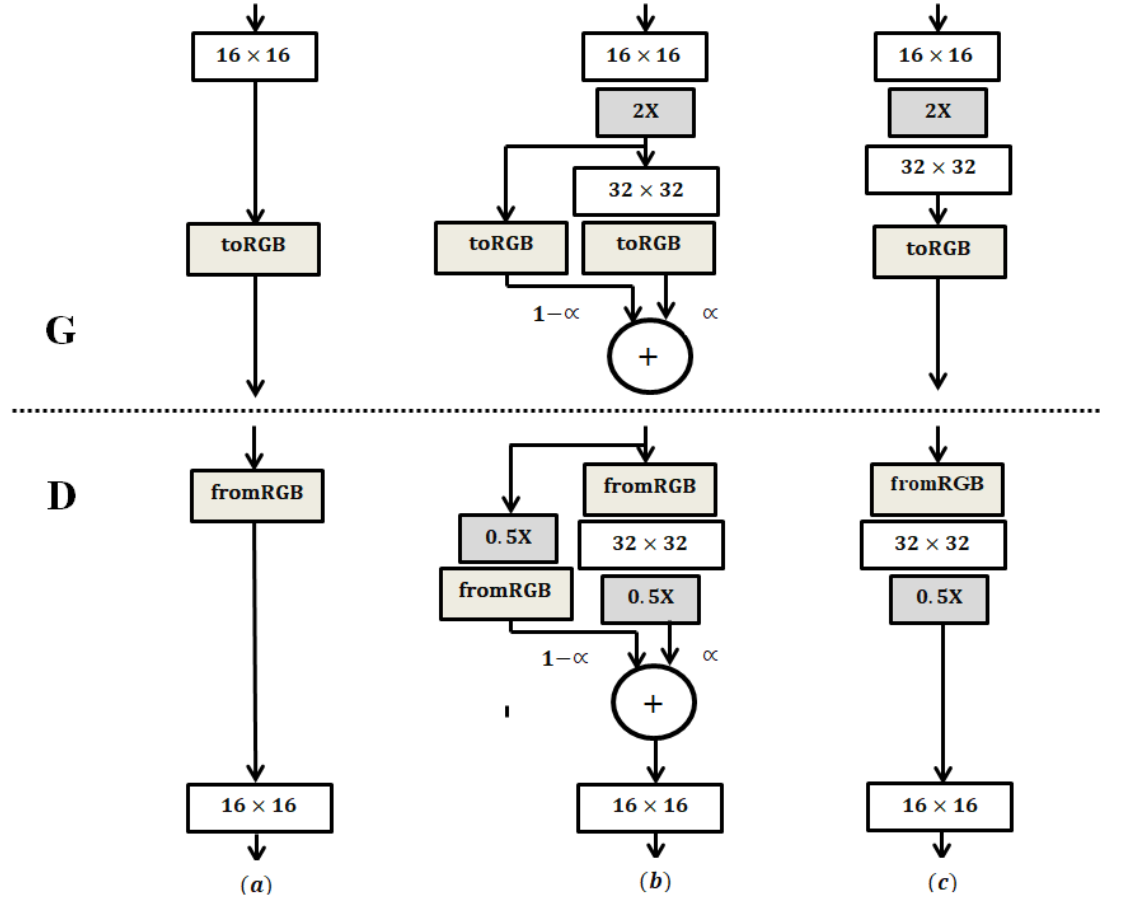


Figure 4.2: Transition from 16x16 images to 32x32 images

In order to achieve this, we created a new model for each pair of models which stacked up the generator on top of the discriminator, allowing the synthetic image to be fed straight into the discriminator model, allowing it to be deemed as real or fake. This model can then be used to train the generator via the discriminator, with the discriminator's weights designated as not trainable and to prevent them from being modified during this process.

This way we created six pairs of composite models, each for the levels of image growth in terms of resolution where each pair consisted of a composite model for the normal or straight-through model and the growth version of the model. This architecture is shown in figure 4.1.

4.3 LOSS FUNCTIONS

4.3.1 Wasserstein Loss

It theoretically demonstrates that GAN training can be gradually optimised. Surprisingly, there is no requirement to balance D and G during training, nor does it necessitate a special network architecture design. The Wasserstein criterion allows us to train D until optimality. When the criterion reaches the optimal value, it simply provides a loss to the generator that we can train as any other neural network. Mode collapse, which is inherent in GANs, can also be minimized using this distance measure. The Wasserstein loss function aims to widen the difference between real and generated image scores.

The following is the Wasserstein loss function, which estimates the distance between the two distributions P_r and P_β :

$$W(P_r, P_\beta) = \sup_{\|f\| \leq 1} \left[E_{z \sim P_r} [f(z)] - E_{z \sim P_\beta} [f(z)] \right] \quad (4.4)$$

Each model in the growth phase of our discriminator network uses Wasserstein loss or in other words, our generator models are trained using Wasserstein loss via the discriminator models.

Since, the WGAN-GP loss can explode due to unavailability of any batch-norm or layer-norm operations in the discriminator, we used the drift penalty with lambda equal to 0.001.

4.3.2 KL Loss Function

KL loss function is used to determine the difference between discrete and continuous probability distributions, where the integral of the events is calculated rather than the sum of the discrete event's probabilities in the latter case. The intuition for the KL divergence score is that when the probability for an event from P is large, but the probability for the same event in Q is small, there is a large divergence. When the probability from P is small and the probability from Q is large, there is also a large divergence, but not as large as the first case.

The negative sum of the probability of each event in P multiplied by the log of the probability of the event in Q over the probability of the event in P is the KL divergence. The divergence for a given event is the value inside the sum. The value of KL divergence loss for two identical distributions is 0.

$$K(\alpha \parallel \beta) = \sum z \ln X P(z) * \log(P(z) / Q(z))$$

(4.5)

We utilized a KL divergence between the Standard Normal Distribution term as well as Conditioning Augmentation Network's output term in generator's loss in order to regulate the latent manifold formed from the encoded text.

CHAPTER 5

RESULTS AND EXPERIMENTATION

The following chapter presents the results obtained from the conducted experiments. Section 5.1 discusses Inception Score metric. Section 5.2 and Section 5.3 depict the variation of discriminator and generator loss respectively with the number of training epochs. Section 5.4 provides a visual representation of the results obtained.

5.1 INCEPTION SCORE

The most important metric to judge the image generated using GANs is Inception Score. Most of the papers that work on GANs use Inception score as its loss function.

Google developed an Image classifier that was a network that based on the image generates a probability distribution for all the available labels for the image. Using this probability distribution we can tell whether the image is distinct (probability of one label is considerably higher than that of other labels).

Inception score uses this idea and generates a score that is a floating point variable ranging from 0 to infinity but in most practical cases this number usually converges to a finite number and describes how close the images generated by the GANs are to a realistic image. It also tells us how much the image looks like a specific label. It thus replaces the human effort to judge the quality of the image.

It also makes use of marginal distribution which is created by first considering some images (say 50,000) and then summing up their label distributions. This helps us to

tell how varied our generated results are. Then we combine both of these scores to generate a new score called Inception score which takes into consideration both aspects that a single image must point towards a single label while all the images must be varied.

A high inception score is obtained when marginal and label distributions are opposite to each other i.e. marginal distributions must be wide and label distributions must be narrow. The higher these two distributions differ, the higher is the value of the inception score. In this report we have calculated the inception score in the form that for each image x , textual encoding z and class label y , the value of marginal distribution $p(y)$ given by:

$$p(y) = \int_x p(y|x = \text{Gen}(z)) dz \quad (5.1)$$

We also used this score for the purpose of checking how our model works and we obtained an IS of 6.86 ± 0.06 over 10 iterations, starting from ten random initializations. Here \pm refers to the standard deviation from the IS calculator. This indicates the quality for our 128x128 resolution images generated using textual embeddings.

5.2 LOSS VARIATION

We trained our network over 6 depths, each depth having 120 epochs with each epoch running over around 32 batches. The plot shown in Figure 5.1 helped us in identifying and diagnosing GAN failures. The figure contains three line sub plots for the discriminator loss for real images (blue), discriminator loss for generated fake images (orange), and the generator loss for generated fake images (green).

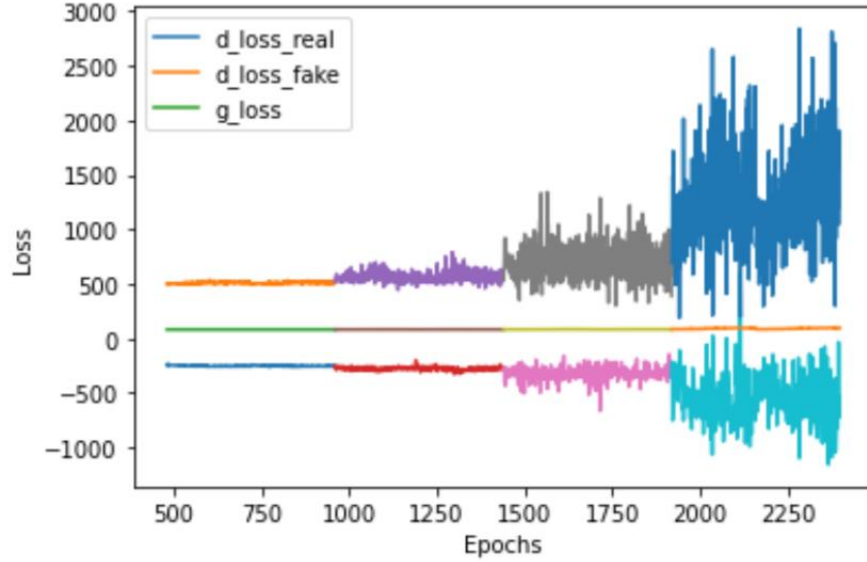


Figure 5.1: Variation of training loss with the number of training epochs

We can see that early in the run, all three losses are a little unpredictable before stabilising about epoch 900 to epoch 1400. After that, the losses stay steady, although the variation increases. If the generator model is capable of producing realistic images, those images are most likely to have been created between the epochs of 1440 and 1920.

These findings are significant because they emphasise the period of stability, and thus, the period during which the generator is projected to provide its best images. It's also worth noting that, even when the training process has stabilized, the quality of images created can and does change throughout the run.

5.3 VISUALIZATION OF RESULTS

This section represents the results obtained with the help of our architecture. Figure 5.2 depicts the visual results obtained from training ProGAN architecture layer-wise whereas figure 5.3 depicts the conversion of textual description to facial image achieved by the model.

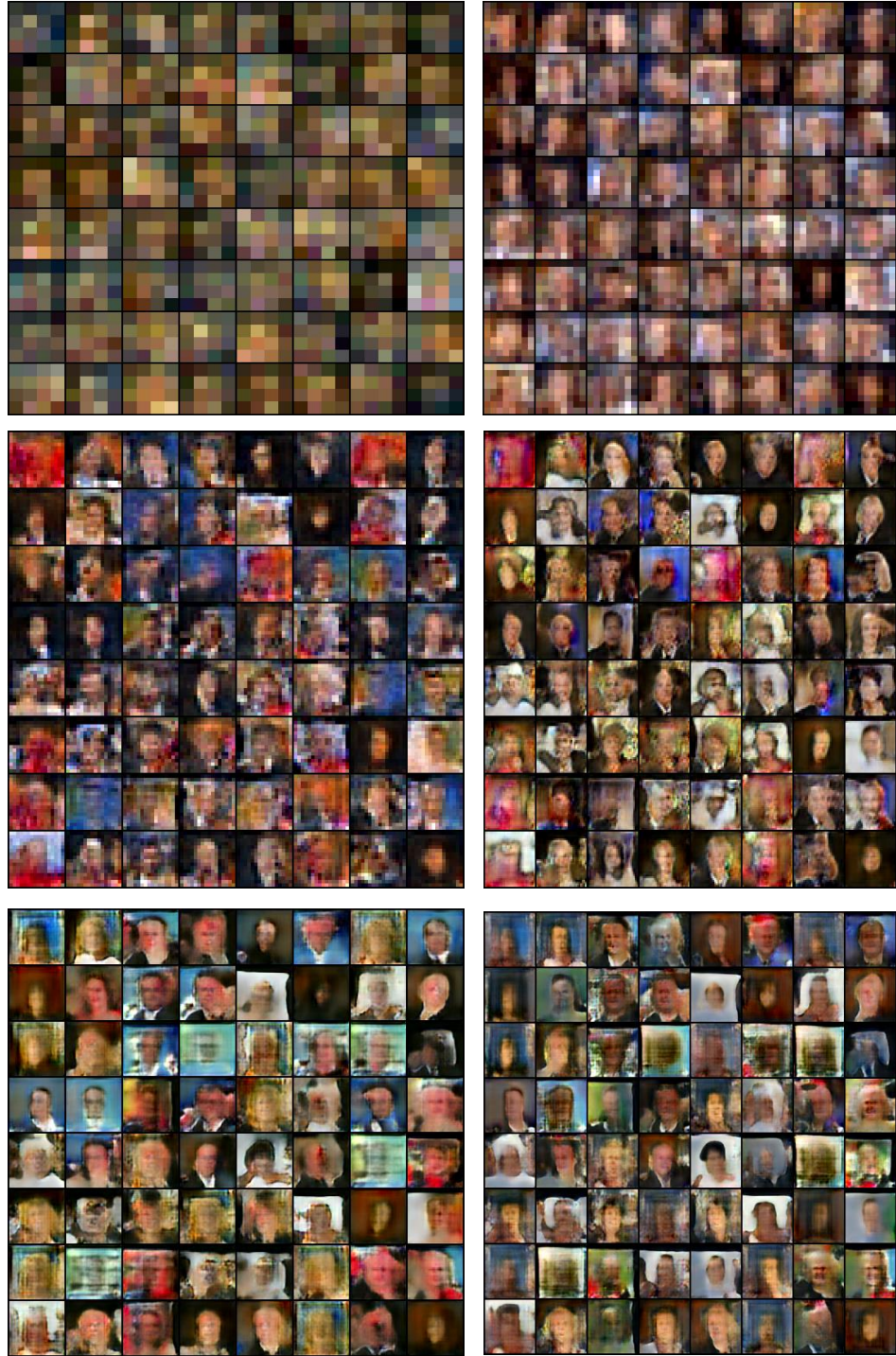


Figure 5.2: Visual results obtained by architecture layer-by-layer

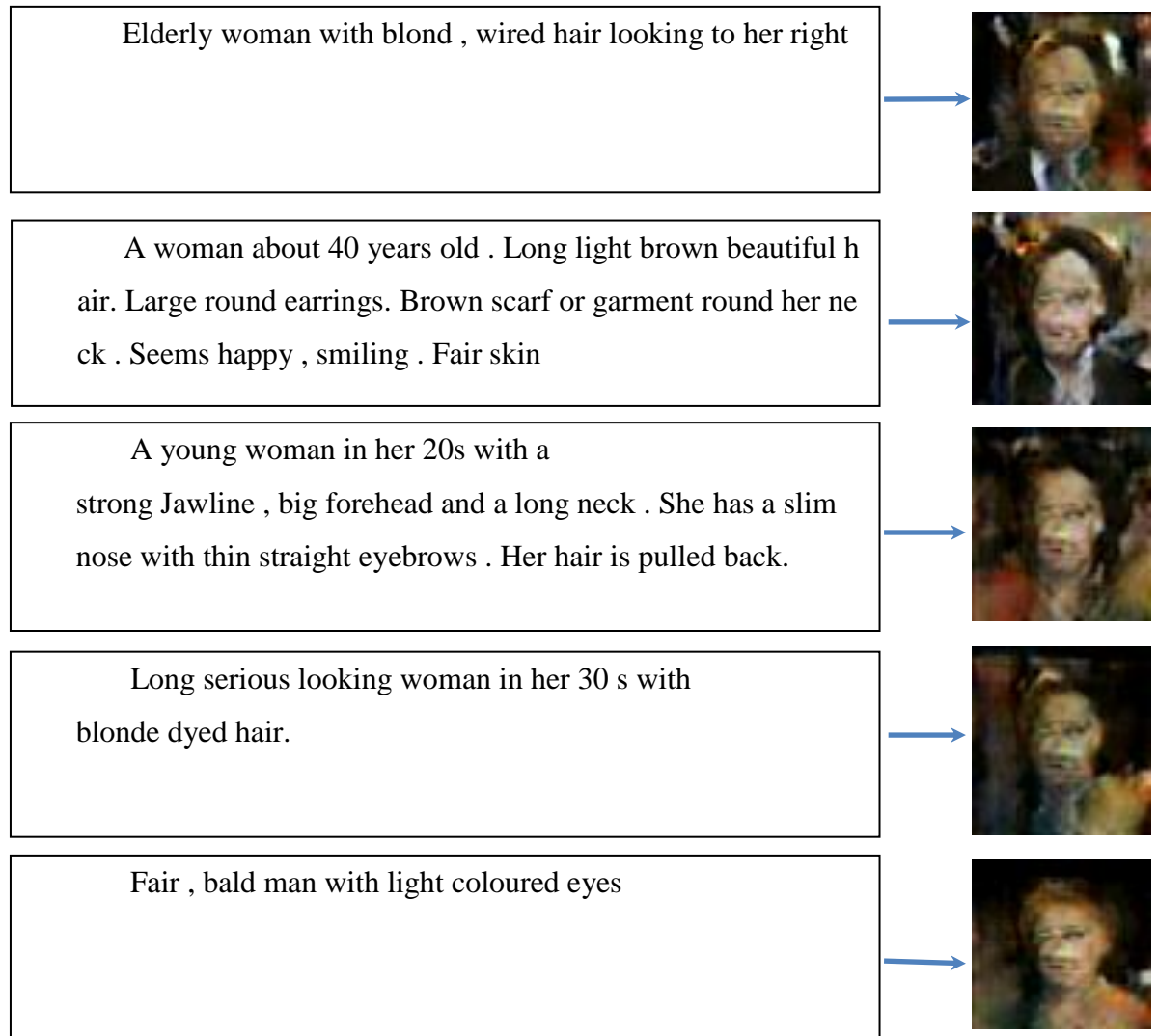


Figure 5.3: Visual results with textual descriptions

CHAPTER 6

CONCLUSION AND FUTURE WORK

The following chapter presents the conclusion arrived at from the conducted research along with prospective directions of future work.

In our work, we have successfully synthesized 128*128 resolution images from textual descriptions using a combined architecture of ProGAN and StackGAN.

The Sentence BERT network is used to convert the textual description into a summary vector. The textual part of the latent vector for the GAN is obtained by passing the summary vector through the Conditioning Augmentation block. Random gaussian noise makes up the second part of the latent vector. The embedding is fed to the final layer of the discriminator for conditional distribution matching, while the latent vector is fed to the generator part of the GAN. The GAN's training advances layer by layer, with higher spatial resolutions. This allows it to create an image with a higher resolution while fine-tuning the details (128*128).

A lot of research is still going on in the field of GANs to handle their existing challenges such as Mode Collapse and Failure to converge. However, we trained a stable GAN and avoided these challenges in our work. One of the major limitations of our work is poor resolution and in future, we can propose an appropriate network for generating more realistic images of higher resolution. Apart from this, it is also important to further enhance the variations in our dataset, not only in terms of a greater variety for textual descriptions, but also for facial images. Also, some other evaluation metrics can be tried out in future in order to better capture the similarity as well as the quality of images obtained.

REFERENCES

- [1] Y. Yu, S. Tang, K. Aizawa, and A. Aizawa, “Category-Based Deep CCA for Fine-Grained Venue Discovery from Multimodal Data,” *IEEE Trans. Neural Networks Learn. Syst.*, 2019, doi: 10.1109/TNNLS.2018.2856253.
- [2] Y. Yu, S. Tang, F. Raposo, and L. Chen, “Deep cross-modal correlation learning for audio and lyrics in music retrieval,” *ACM Trans. Multimed. Comput. Commun. Appl.*, 2019, doi: 10.1145/3281746.
- [3] I. J. Goodfellow *et al.*, “Generative adversarial nets,” 2014, doi: 10.3156/jsoft.29.5_177_2.
- [4] A. Van Den *et al.*, “Conditional Image Generation with PixelCNN Decoders,” *Adv. Neural Inf. Process. Syst.*, 2016.
- [5] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” 2014.
- [6] A. Makhzani and B. Frey, “PixelGAN autoencoders,” 2017.
- [7] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, “CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training,” 2017, doi: 10.1109/ICCV.2017.299.
- [8] J. Gauthier, “Conditional generative adversarial nets for convolutional face generation,” *Tech. Rep.*, 2014.
- [9] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text to image synthesis,” 2016.
- [10] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” 2016.

- [11] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, “Learning what and where to draw,” 2016.
- [12] H. Dong, S. Yu, C. Wu, and Y. Guo, “Semantic Image Synthesis via Adversarial Learning,” 2017, doi: 10.1109/ICCV.2017.608.
- [13] S. Hong, D. Yang, J. Choi, and H. Lee, “Inferring Semantic Layout for Hierarchical Text-to-Image Synthesis,” 2018, doi: 10.1109/CVPR.2018.00833.
- [14] S. Sharma, D. Suhubdy, V. Michalski, S. E. Kahou, and Y. Bengio, “ChatPainter: Improving text to image generation using dialogue,” 2018.
- [15] H. Zhang *et al.*, “StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks,” 2017, doi: 10.1109/ICCV.2017.629.
- [16] H. Zhang *et al.*, “StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019, doi: 10.1109/TPAMI.2018.2856256.
- [17] T. Xu *et al.*, “AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks,” 2018, doi: 10.1109/CVPR.2018.00143.
- [18] T. Qiao, J. Zhang, D. Xu, and D. Tao, “Mirrorgan: Learning text-to-image generation by redescription,” 2019, doi: 10.1109/CVPR.2019.00160.
- [19] Z. Zhang, Y. Xie, and L. Yang, “Photographic Text-to-Image Synthesis with a Hierarchically-Nested Adversarial Network,” 2018, doi: 10.1109/CVPR.2018.00649.
- [20] L. Gao, D. Chen, Z. Zhao, J. Shao, and H. T. Shen, “Lightweight dynamic conditional GAN with pyramid attention for text-to-image synthesis,” *Pattern Recognit.*, 2021, doi: 10.1016/j.patcog.2020.107384.

- [21] M. Wang, C. Lang, S. Feng, T. Wang, Y. Jin, and Y. Li, “Text to photo-realistic image synthesis via chained deep recurrent generative adversarial network,” *J. Vis. Commun. Image Represent.*, 2021, doi: 10.1016/j.jvcir.2020.102955.
- [22] Y. Dong, Y. Zhang, L. Ma, Z. Wang, and J. Luo, “Unsupervised text-to-image synthesis,” *Pattern Recognit.*, vol. 110, p. 107573, Feb. 2021, doi: 10.1016/j.patcog.2020.107573.
- [23] H. Zhang *et al.*, “Cross-Modal Contrastive Learning for Text-to-Image Generation.”
- [24] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” 2018.
- [25] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” 2019, doi: 10.1109/CVPR.2019.00453.
- [26] Y. Choi, M. Choi, M. Kim, J. W. Ha, S. Kim, and J. Choo, “StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation,” 2018, doi: 10.1109/CVPR.2018.00916.
- [27] Y. Lu, Y. W. Tai, and C. K. Tang, “Attribute-guided face generation using conditional cycleGAN,” 2018, doi: 10.1007/978-3-030-01258-8_18.
- [28] M. E. Nilsback and A. Zisserman, “Automated flower classification over a large number of classes,” 2008, doi: 10.1109/ICVGIP.2008.47.
- [29] P. Welinder *et al.*, “Caltech-UCSD Birds-200 CNS-TR-2010-001,” California Institute of Technology, 2010. Accessed: Feb. 15, 2021. [Online]. Available: <https://resolver.caltech.edu/CaltechAUTHORS:20111026-120541847>.
- [30] X. Chen *et al.*, “Microsoft COCO Captions: Data Collection and Evaluation Server.”

- [31] X. Chen, L. Qing, X. He, X. Luo, and Y. Xu, “FTGAN: A fully-trained generative adversarial networks for text to face generation,” *arXiv*. 2019.
- [32] M. Z. Khan *et al.*, “A Realistic Image Generation of Face from Text Description using the Fully Trained Generative Adversarial Networks,” *IEEE Access*, pp. 1–1, Aug. 2020, doi: 10.1109/access.2020.3015656.
- [33] T. Wang, T. Zhang, and B. C. Lovell, “Faces la Carte: Text-to-face generation via attribute disentanglement,” *arXiv*. 2020.
- [34] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” 2020, doi: 10.1109/CVPR42600.2020.00813.
- [35] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” 2014, doi: 10.1007/978-3-319-10602-1_48.
- [36] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, “Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models,” *Int. J. Comput. Vis.*, 2017, doi: 10.1007/s11263-016-0965-7.
- [37] D. Elliott and A. P. De Vries, “Describing images using inferred visual dependency representations,” 2015, doi: 10.3115/v1/p15-1005.

APPENDIX A

PUBLICATION DETAILS

Paper Title: Realistic face generation using a textual description

Author Names: Anukriti Kumar, Anurag Mudgil, Nakul Dodeja, Dinesh Kumar Vishwakarma

Name of Conference: 5th International Conference on Computing Methodologies and Communication (ICCMC 2021)

Conference Date and Venue: 08-10th April, 2021 (Surya Engineering College, Erode, Tamil Nadu)

Status of the paper: Accepted and published

Date of paper acceptance: 20th February, 2021

Date of publication: 6th May, 2021

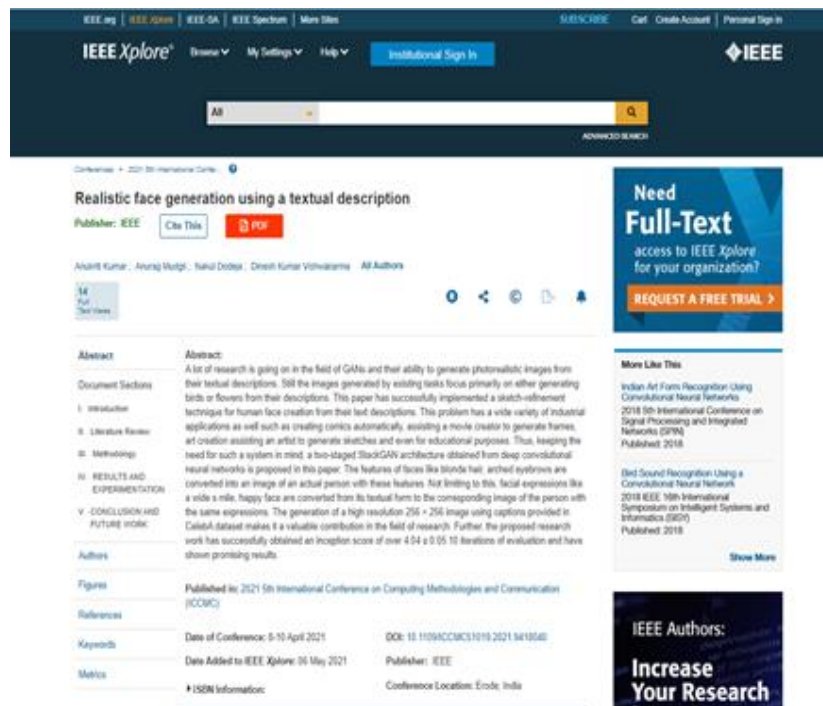


Figure A.1: Proof of publication