FACE GENERATION FROM TEXTUAL DESCRIPTION

BY

ABDULLAH AL ABID
ID: 161-15-7429

AND

TOUFA RANI SAHA
ID: 161-15-7517

AND

MD. ASHAB BIN RAYHAN
ID: 161-15-7084

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science & Engineering.

Supervised By

Shah Md. Tanvir Siddique
Assistant Professor
Department of Computer Science and Engineering
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY
DHAKA, BANGLADESH

DECEMBER 2019

APPROVAL

This Project/internship titled **"FACE GENERATION FROM TEXTUAL DESCRIPTION"**, submitted by Abdullah Al Abid, ID No: 161-15-7429, Ashab Bin Rayhan, ID No: 161-15-7084 and Toufa Rani Saha, ID No: 161-15-7517 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 07/12/2019.

BOARD OF EXAMINERS

Dr. Syed Akhter Hossain — **Chairman**
Professor and Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Abdus Sattar — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Shaon Bhatta Shuvo — **Internal Examiner**
Senior Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Dr. Md. Saddam Hossain — **External Examiner**
Assistant Professor
Department of Computer Science and Engineering
United International University

DECLARATION

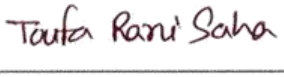
We hereby declare that this project has been done by us under the supervision of **Shah Md. Tanvir Siddique, Department of Computer Science and Engineering**, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

Shah Md. Tanvir Siddique
Assistant Professor
Department of Computer Science and Engineering
Daffodil International University

Submitted by:

Abdullah Al Abid
ID: 161-15-7429
Department of Computer Science and Engineering
Daffodil International University



Toufa Rani Saha
ID: 161-15-7517
Department of Computer Science and Engineering
Daffodil International University

Md. Ashab Bin Rayhan
ID: 161-15-7084
Department of Computer Science and Engineering
Daffodil International University

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratitude to almighty Allah for His divine blessing makes us possible to complete this project successfully.

We feel grateful to and wish our profound our indebtedness to **Shah Md. Tanvir Siddique, Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University, Dhaka. Deep knowledge & keen interest of our supervisor in the field of machine learning and image processing to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior draft and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to **Prof. Dr. Syed Akhter Hossain, Professor & Head**, Department of Computer Science and Engineering, for his kind help to finish our project and also to other faculty members and the staff of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discussion while completing the course work.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

ABSTRACT

The main object of this project is to generate a human face based on a given description. Generating images from text plays a huge role for public safety. This is a sub domain of text-to-image synthesis. This project is a part of deep learning. This project uses two latest architectures for generating images from text. StackGAN and ProGAN for synthesis of the image from the given description. We are using a dataset named ‘Face2Text’, which contains 400 facial images and textual captions for each of the image. The key idea is to grow both the generator and discriminator progressively: starting from a low resolution, we add new layers that model increasingly fine details as training progresses. The final generated output is not that high quality but comparable with the given input.

Keywords: text-to-face generation, tensorflow-gpu, pytorch-gpu, ProGAN, StackGAN, Deep learning

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1Introduction

Generate images from text descriptions is an amazing demonstration of deep learning. Text-to-image synthesis helps us to mining the relationship between text and image. It's aiming to generate natural images from input descriptions. The classification tasks such as sentiment analysis have been successful with deep neural networks that are understand to learn discriminative vector representations from description which is given by user or human. Here, we take text from user written descriptions and convert it image pixels. Description may about any object, bird or human face. We worked on the description about human face like, "a woman has big eyes", "golden curly hair", "curve eyebrows" and "flatted face". It is a challenging task. To solve this problem, we need also solving two sub problems that is first, learn a text analysis feature representation that captures the important details from text and second, use this features to synthesize a complete image that a human wants for real. Fortunately, deep learning has the capacity to solve the challenge. In this project, we try our best to give our highest effort to implement a simple and effective GAN architecture and training strategy, which is enable to making image of a human face from text descriptions.

In this paper, we introduce an automated system integrated with GAN technology that will give us like a real image, which will help us for public security. First, we write descriptions as input in our system. The text input will be processed by using GAN algorithm and it will reply with the desired output, which is an image of a human face.

1.2 Motivation

Now is the time of technological revolution. In today's digital age, it is the most important thing, that we ensure our security. Day by day, we go through the era of digitalization and crimes are increased. We see that if any occurrences will occur and there was any eye witness then police try to make a sketch of the suspect by the help of witness given him/her

descriptions, which has made a positive impact on the output of the project. We thought that if it is possible to make sketch from descriptions in computer then it is quite easy to solve the case. However, was not so easy task. Another example, when we read novels, books etc. then we imagine the scenery and get happiness, but if we see the scenery or characteristics of novel in front of our eyes then our excitement going twice undoubtedly. This concept also motivated us to doing this type of interesting things.

Deep learning theories have been applied in our study for converting text to image. For better output, we use Pro-GAN, which is also a deep learning algorithm and updated algorithm of GAN.

1.3 Rationale of the Study

I have always been excited while reading storybooks how the characters mentioned in the books would look in reality. However, imagining it is still visible, but if I want to get details to the description is quite challenging. Many times, I end up imagining a very blur face for the character until the very end of the story. It is only when the book is translated into a movie that the blur face is filled up with details. For instance, I could never imagine the exact face of Rashed from the book 'Amar Bondhu Rashed'. However, when the movie came out, I try to relate with Chowdhury Zawata Afnan's face of Rashed. The casting professionals must work very hard to get the exact character based on written in the script.

This problem inspired me to find a solution for it. Then we try to start a search from the deep learning research literature. In addition, we find something similar from it. Fortunately, there we did a huge research for creating images from text.

1.4 Research Questions

In order to have reasonable, accurate and realistic response to the problem, the researcher wishes to pose the following questions to enable him to examine the identified problem.

- How does the suspects face look like?

1.5 Expected Output

- Generates image of the suspects face from description.

1.6 Report Layout

There are five chapters in this research paper. They are Introduction, Background, Research Methodology, Experimental Results and Discussion, AND Summary, Conclusion, Recommendation and Implication for Future Research.

Chapter 1: Introduction; Introduction, Motivation, Rationale of the Study, Research Questions, Expected Output, Report Layout.
Chapter 2: Background; Introduction, Related Works, Research Summary, Scope of the Problem, Challenges.
Chapter 3: Research Methodology; Introduction, Research Subject and Instrumentation, Data Collection Procedure, Implementation Requirements.
Chapter 4: Experimental Results and Discussion; Introduction, Experimental Results, Descriptive Analysis, Summary.
Chapter 5: Summary, Conclusion, Recommendation and Implication for Future Research; Summary of the Study, Conclusions, Implication for Further Study.

CHAPTER 2
BACKGROUND

2.1 Introduction

Deep learning is an important part of machine learning chapter based on artificial neural network. Deep learning uses multiple layers, which increasingly detect higher-level features from the raw given input. We see that, in image processing, lower layers identify edges but in higher layers which identify the concepts similar to a human such as letters or faces.

In our project, we use deep learning algorithms. We use GAN model to implement this concept. For better output or for more realistic image we apply Pro-GAN model in our project. Lan Goodfellow invented GAN (Generative Adversarial Network) model, which is a class of machine learning systems. Here two neural network used. The generative networks generate candidates. Second, one, which is discriminative network, evaluates candidates. A known dataset given as the initial training data for the discriminator. Then the discriminator evaluates candidates synthesized by the generator. The generator produces better images.

2.2 Related Work

ProGAN: People of the high resolution images looks like real, but actually they are not. They were synthesized by ProGAN. ProGAN is a type of generative adversarial network. It was published by Keras. Progressive Growing of GANs uses for improving quality, stability and variation. Here, we use ProGAN for better image quality. In GAN we work low resolution images, but in ProGAN it is possible to gradually improved the resolution by combining layers to the networks. In ProGAN we also use generator and discriminator. All the layers in both networks remain trained in all respects the training process. A detailed diagram of working procedure of ProGAN is provided in Appendix A.1 .

-

StackGAN: StackGAN is also applied for Text to Photo-Realistic image synthesis. Generative adversarial networks (GAN) is the most fascinating idea in the last few years in machine learning. Text to photo realistic image synthesis with Stacked generative adversarial networks which provides a deep learning architecture which capable of making realistic images from the given description. StackGAN has two-stage network. Stage one, generates (64 * 64) images, structural information and low details. Stage one of StackGAN makes rough shapes images and basic color of objects. Stage two, needs stage one outputs, higher detail, photo realistic, samples (256 * 256). Both stages take in the same textual input. Stage two, able to identify defects in stage one results and adding details with the refinement process. Stage two, again read the text description to complete details of the object. Then producing a higher resolution photo-realistic image.

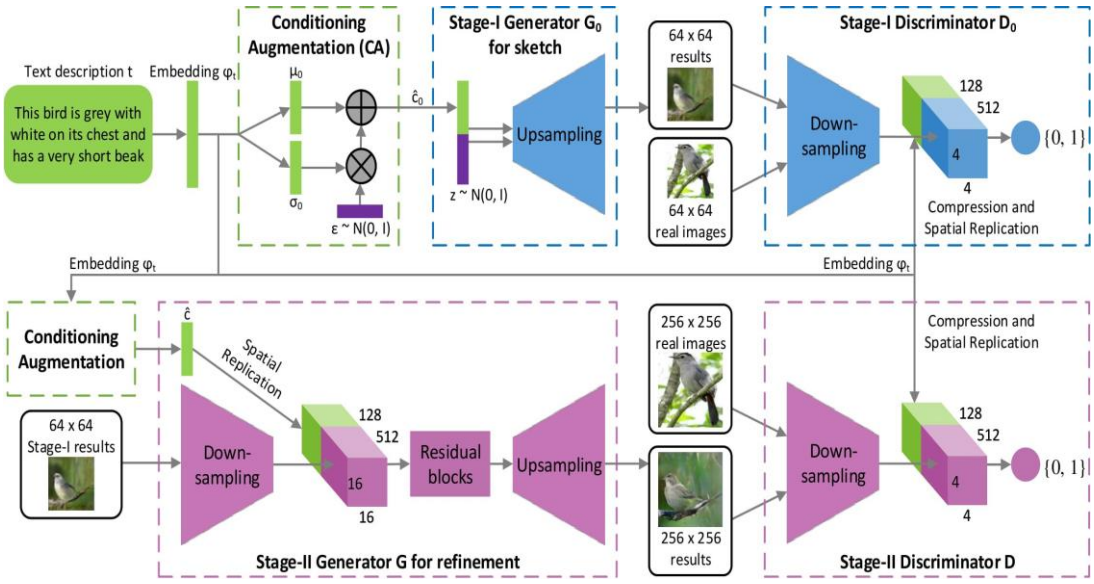


Figure 2.1: Stack GAN architecture. [3]

FAGAN: FAGAN means Full Attention GAN. FAGAN is the part of SAGAN (Self Attention GAN). The SAGAN architecture just adds one self-attention layer to the generator and one to the discriminator of the DCGAN architecture. Besides, for creating the **Q**, **K** and **V** feature banks for self-attention, the layer uses (**1 x 1**) convolution. Full Attention GAN with figure 2.2. From this figure, we see that here two paths are available;

on the upper path we compute traditional convolutional output. In addition, the lower path, we have an attention layer which generalizes to (k*k) convolutions filters instead of (1*1) filters.

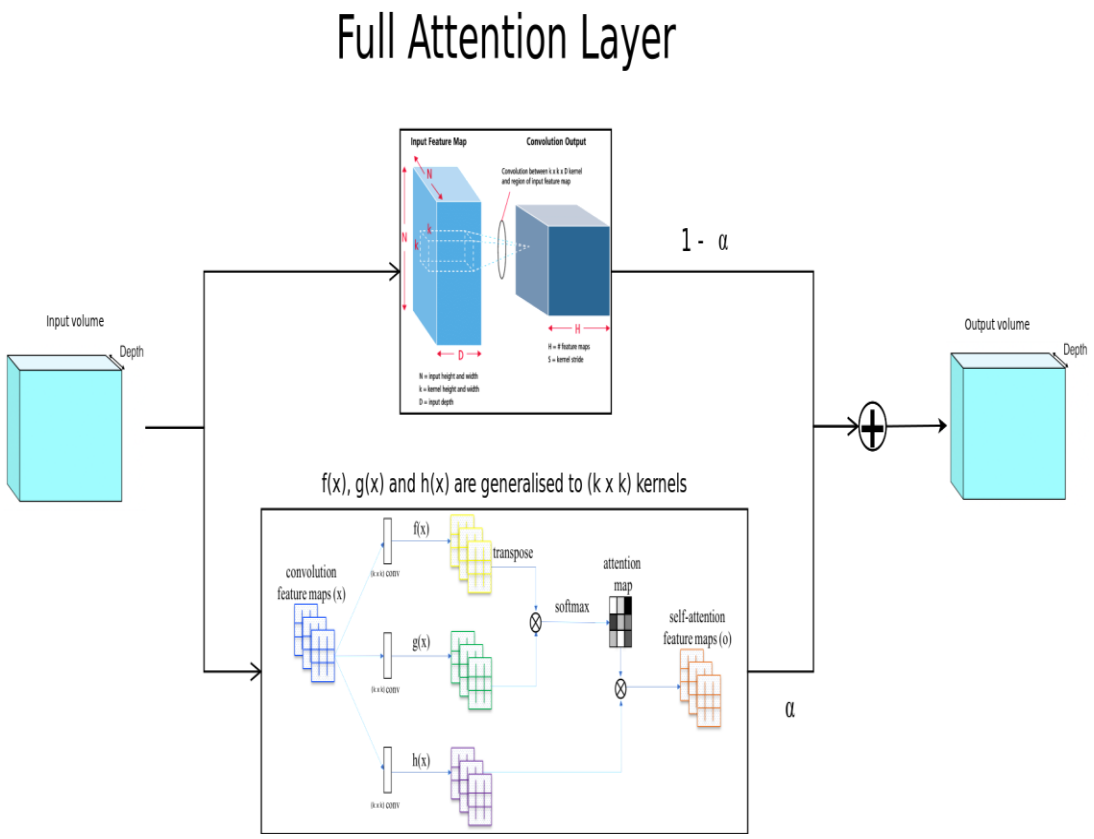


Figure 2.2: Full Attention GAN Layer. [4]

The application of deep learning techniques has increased broadly in the world in the last few years, especially in the protection of human life. However, there are not many applications that offer portable solution that enables to create text to image using deep learning techniques. Few researchers described different implementations of making realistic images using deep learning algorithms also.

The paper finds that the development of machine learning sector has been done widely and shows how solutions can be created using deep learning technology that will help in addressing some of the problems. Researchers developed a system, which capable to create images from description. Nowadays, people research in this topic widely and few

researchers develop more deep learning algorithms to achieve better image or more realistic image. After GAN, many researchers use StackGAN, ProGAN, FAGAN etc. for getting better output. We think that our output is not so bad when we implementing GAN and ProGAN in our project. In the future, we will work on it for better achievement.

2.3 Research Summary

Implementation of GAN algorithm, collecting dataset and give description input of Text to image generation provide the basis for what is to be done and who is to do it in order to reduce the risk of low quality images. The goal is generating images from text using user given description. For example, specific exposure sources can be linked to specific technologies; it will help to create high quality images, which is necessary to control the risk.

2.4 Scope of the Problem

- The images are not too much clear as like realistic images.
- Training model with large dataset.
- This prototype needs high quality graphics cards.

2.5 Challenges

The main challenge towards us was collecting dataset on different characteristics images. Then we faced some difficulties to integrate tensor flow trained model with pytorch and make it working. As we did not work with web framework before it was difficult to make it working. The output images are not looks more real.

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Introduction

Generating an imaginary image from the detailed description of the face is a very interesting thing. However, the whole process is a little bit challenging. The first problem is getting proper dataset. Besides our main architecture for the project was to collect facial features or details from the textual description and by using them generating an image. At the beginning, we thought to use NLP for the textual description part in our project, but using NLP is another complex method and combining it with our GAN technology is another complex procedure. That is why we used some pretrained text encoder and some pretrained conditioning augmentation. As a result, our problem to handle the textual description part is solved and we have done it only by our GAN method. Here we have used two GAN architecture. One is StackGAN and the other one is ProGAN. Where StackGAN is used to handle the text encoding and conditioning augmentation part and the ProGAN handles the image synthesis part.

3.2 Research Subject and Instrumentation

As our target is to generate a human face using ProGAN and StackGAN then we used Python to use it effectively and make the best use of it. As we mentioned earlier, StackGAN is used for text encoding with and the ProGAN is used for the image synthetization. The main StackGAN method generally uses multiple GANs. It uses those GANs at different resolutions. That's why StackGAN is a perfect technique for the distributed matching problem. Besides this, ProGAN only use one GAN technique and it is trained progressively step by step. Therefore, we have combined these two methods to train our model for generating images with the Face2Text dataset, which contains 400 images of celebrities.

3.3 Data Collection Procedure

Collecting the dataset was difficult by any stretch of the imagination. We were searching for a proper dataset which will perform well with our task. In the interim some time passed, and this exploration approached Face2Text: Collecting an Annotated Image Description Corpus for the Generation of Rich Face Descriptions: exactly what we needed.

The Face2Text v0.1 dataset contains regular language portrayals for 400 randomly selected pictures from the [7] LFW (Labeled Faces in the Wild) dataset. The portrayals are cleaned to expel hesitant and superfluous subtitles accommodated the individuals in the pictures. A portion of the depictions portray the facial highlights, yet in addition give some suggested data from the photos. For example, one of the subtitles for a face peruses: "The man in the image is likely a criminal". Because of every one of these elements and the moderately smaller size of the dataset, we chose to utilize it as a proof of idea for our design.

3.4 Statistical Analysis

For analyzing our model architecture, we generate some loss values while training our GAN model. Three types (d_loss, g_loss, kl_loss) of loss were generated during training process. ‘d_loss’ indicates the discriminator loss while training the discriminator and ‘g_loss’ indicates generator loss while the training the image generator. ‘kl_loss’ indicates the Kullback-Leibar Divergence. Which is also known as relative entropy. This measures in which manner one value is different from the other and reference probability distribution.

We have tried to analyze the bar graph and stem plot of our generated loss by plotting ‘d_loss’, ‘g_loss’, ‘kl_loss’ with python matplotlib library.

3.4.1 Bar graph of loss data:

Code to generate bar graph of loss data using matplotlib is provided in Appendix C.1.

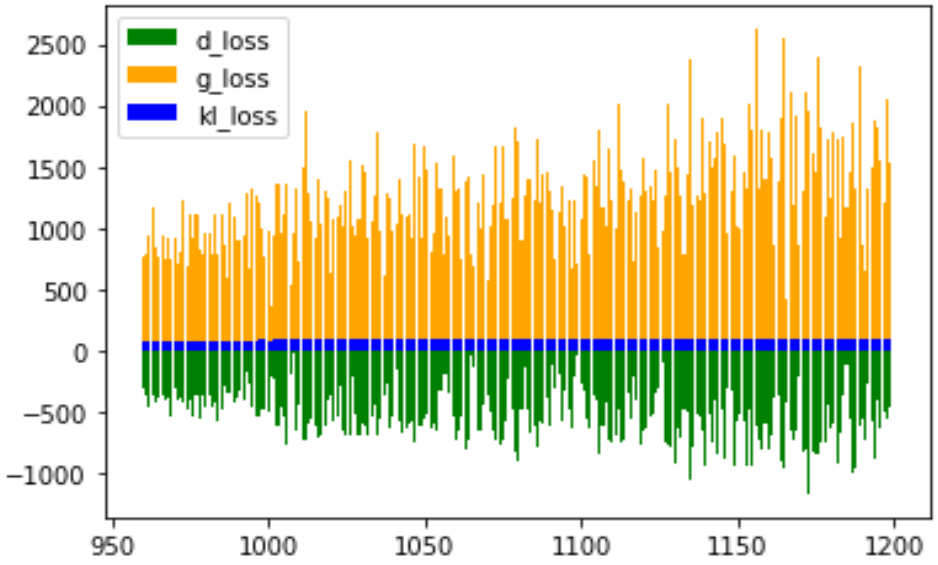


Figure 3.1: Bar Graph of loss data for the Trained Data

3.4.2 Line Plot of loss data:

Code to generate Line Plot of loss data using matplotlib is provided in Appendix C.2.

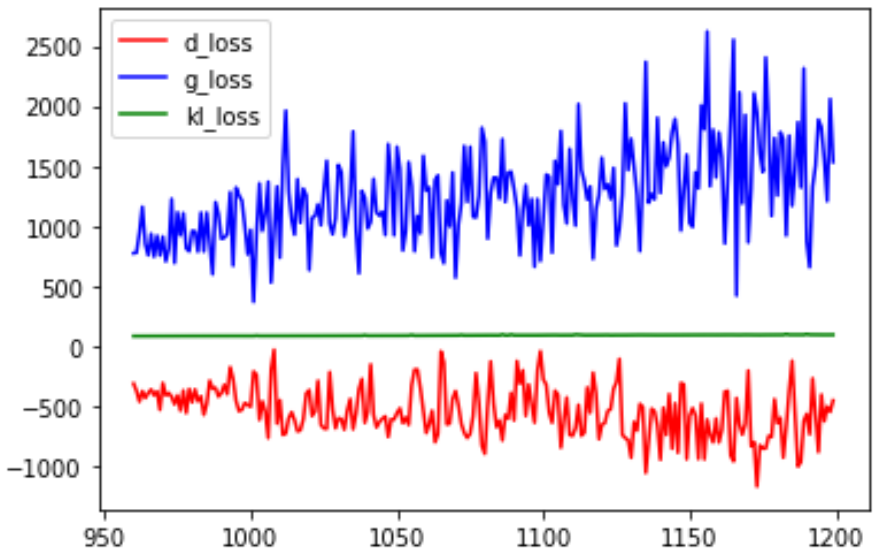


Figure 3.2: Line Plot diagram of loss data for the Trained Data

3.5 Implementation Requirements

The fundamental necessity for our task is Python, PyTorch and Tensorflow. The design was actualized in Python utilizing the PyTorch structure. We have worked with tensorflow and keras prior thus We wanted to attempt PyTorch once. We truly loved the utilization of a python local debugger for investigating the Network Architecture; a cordiality of the excited execution technique. Tensorflow has as of late incorporated an enthusiastic execution mode as well. Anyway, this isn't a discussion on which structure is better, we simply needed to feature that the code for this network has been written in PyTorch.

This project also requires some runtimes like -

- **easydict** - allows the directory values to use as an attribute
- **pillow** - this is and imaging library which allows to open, manipulate and save image
- **PyYAML** - allows YAML parsing which makes easy readability of the scripted languages to the human
- **numpy -** this allows scientific operations in Python and a basic package of Python.

These features are obvious for this project.

We used **'Google Colab'** for implementing this project. This project requires high end gpu and ram to train the model and generates lots of trained data. So, we used **'Google Colab'** as it has hosted high quality gpu and ram.

3.5.1 Training the GAN (Generative Adversarial Network)

All the latest GAN architectures uses photos with textual captions embedded in each of the photo. Training them with just textual face description and then match with the preloaded text encoder is really a challenging task. This task requires high end GPU running machines like NVIDA GTX1070. Otherwise the training process will run slow and will run out of GPU cache memory. In this project, we will reuse the previously trained encoder and GAN model from LFW [7] image dataset trained on 'Google Colab' and simply train a new model to generate the desired face.

Though the generated output is not as good as we expected as the training the model is not an easy task. It consumes a lot of time and GPU resources. Due to lack of big dataset the trained model cannot generate high resolution images. We only used 400 images of random celebrities with labeled data. The training process can take 10 or more hours to train the model for a better output. Mainly GAN model is developed by NVIDIA company and they suggest to use high end GPU (minimum requirement is: GTX 1070 or higher). But high-end GPU is too much costly. At the beginning we tried with regular GPU computers but we failed. The training stopped and showed CUDA out of memory error. CUDA memory couldn't handle the bigger cache file of our training process. This can be only resolved by reducing training batch size which generates an incomplete trained model. That's why we switched to 'Google Colab' which is developed by Google based on jupyter notebook. The detailed specifications of 'Google Colab' is provided in Appendix B.1.

Once the GPU problems are solved, the actual training process of the top layer of the network begins. You'll see a series of different outputs,

- **losses:** In this section losses results produced by our two neural networks is saved.

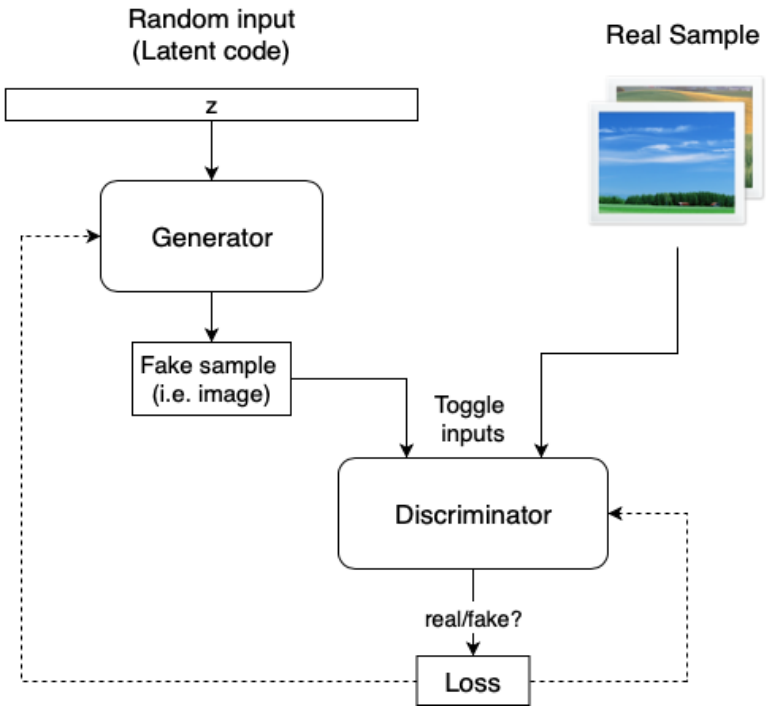


Figure 3.3: Architecture of loss generation [6]

- **saved_models** : In this section different steps of trained models are saved as a .pth file format.
- **generated_samples**: In this section our GAN model produces pixelated images of some given captions and images.

Figure 3.4: Real images of sample training data

young athletic female showing off medal and looking fairly satisfied with herself .

long faced early 5 0 s man , with gray and very short hair , wearing formal clothes . his eyes are small , long nose and normal mouth . he seems to be talking about something serious . has a few wrinkles and his face is thin

mostly bald man in his 6 0 s , with just some light hair on the side of his face . he seems to be giving a presentation or speaking to an audience . he is wearing spectacles with a thin frame but big lenses , and looks smart (wearing a suit and tie) . he has light eyes and seems to have most of his teeth . he has wrinkles starting from his cheek downwards to his neck .

fair skinned woman , approximately late 2 0 s , early 3 0 s . light hair is tied back into an updo . she has got a slim nose and a wide smile .

pale female , hair pulled up . she has makeup and a lovely smile

slightly dark skinned young male . possibly in his 2 0 s and of african - american race . short dark brown hair with thick eyebrows . brown eyes , average sized nose and clean shaven . plain face with average forehead size . looks motivated / determined / foused / on edge / tense / angry .

a man in his late 5 0 s , full head of gray hair , small eyes and eyebrows and lips . wide chin . his expression seems serious and not amused . kind of inquisitive

a young woman carrying an old style of hair which is sleek pulled out of her face and with curls in front of her ears . blue eyed , wide stare , smiling and full cheeks . wearing a necklace

a middle aged lady with dark skin . she has a big fore head and a big nose . she is smiling . her eyes are wide apart .

the man looks middle - aged . his forehead is shiny and his nose is rather large . his hair is blonde , shaven on the sides of his head with a tuft at the front of his head . he looks like he ' s in the middle of saying something .

determined - looking woman , in the early thirties , dark hair and eyes , suntanned , possibly an athlete .

Figure 3.5: Text captions of real images used in training process

We trained quite a few versions using different hyper parameters. As alluded in the prior section, the details related to training are as follows:

- As the discriminator doesn't use any batch-norm or layer-norm operations, the WGAN-GP loss (used here for training) can explode. That's why the drift penalty with lambda = 0.001 is used here.
- For controlling the dormant complex made from the encoded content, we have to utilize a KL dissimilarity (between CA's yield and Standard Ordinary circulation) term in Generator's loss.
- As the generated images is not so good according to the descriptions we need to use WGAN variant of the Matching-Aware discriminator to make the generated images conform better to the input textual distribution.

- The fade-in time for higher layers should be more than the fade-in time for lower layers. To determine this, we utilized a rate (85 to be exact) for fading in new layers while preparing

- Due to the insufficient amount of data (only 400 images), there is a huge problem we found in our generated output image. The problem is it creates some blurry background on the higher resolution images but this problem was not found on the lower resolution output image.

- The training process takes lots of time to train the model perfectly. So, we need to utilize the time by changing the hyper parameters of the trainer. We tried to use bigger epoch number for lower depths and smaller epoch number for the higher depth levels.

By default, this script will run 1200 training steps. Each step produces single image at random from the training set, our model randomly selects images from the dataset to generate a single sample image. Our model is trained by a config file where details of hyper parameters are written.

Full details of our hyperparameters (cofig file) to train our model is provided in Appendix C.4.

Here are some images for understanding the image quality on different depth levels:

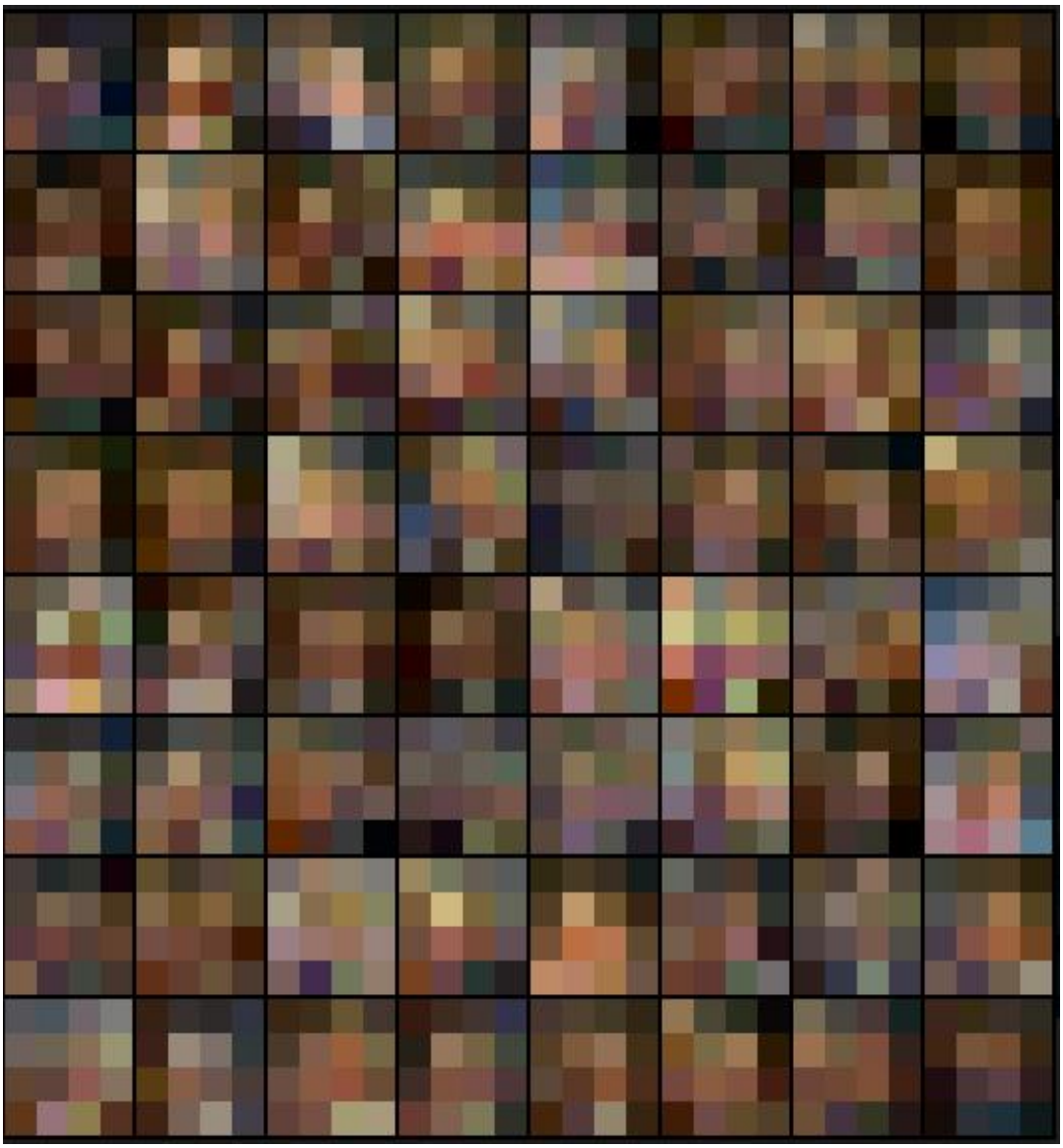


Figure 3.6: Generated Image during training process

[depth 1, epoch 120]

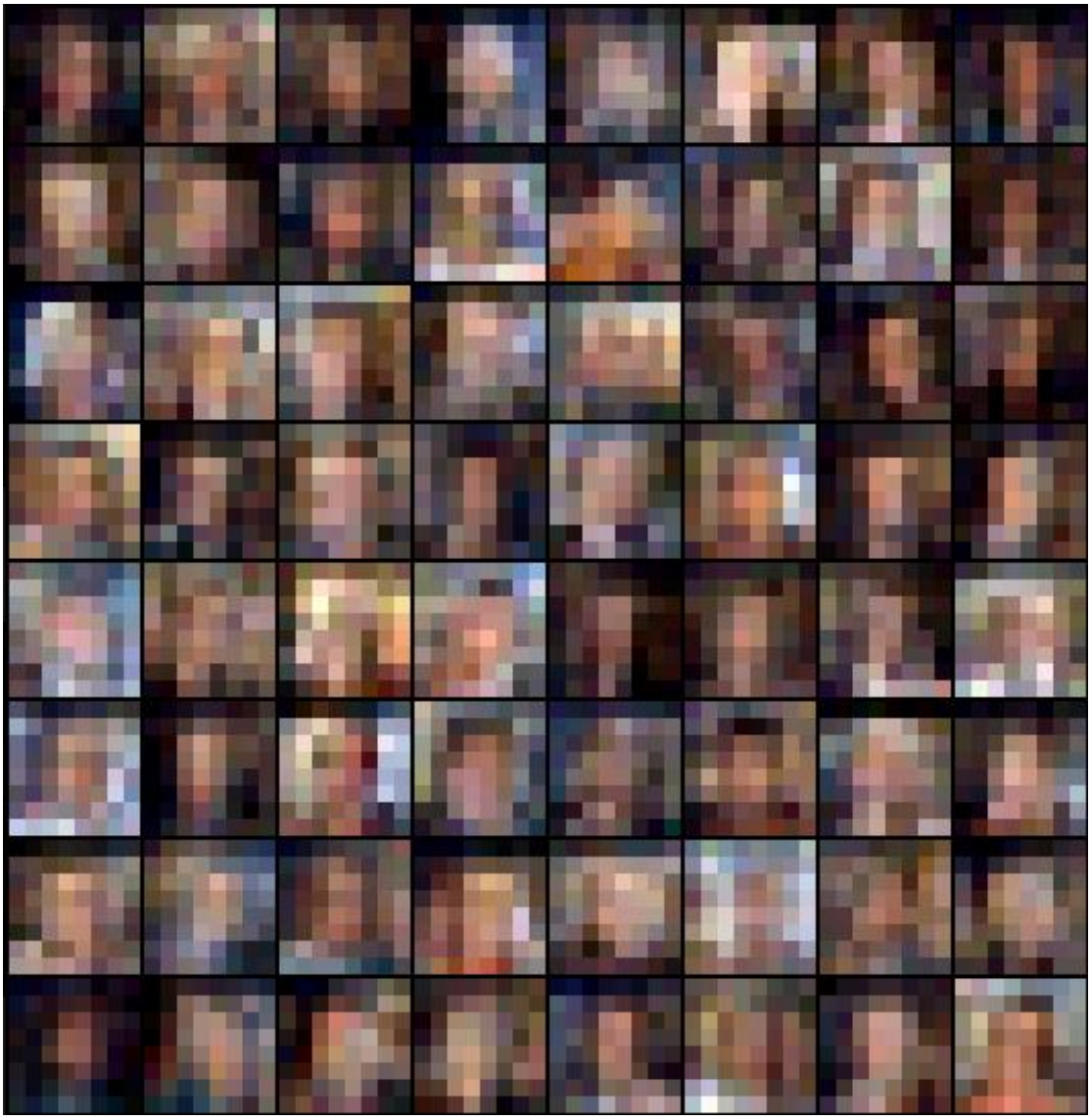


Figure 3.7: Generated Image during training process
[depth 2, epoch 120]

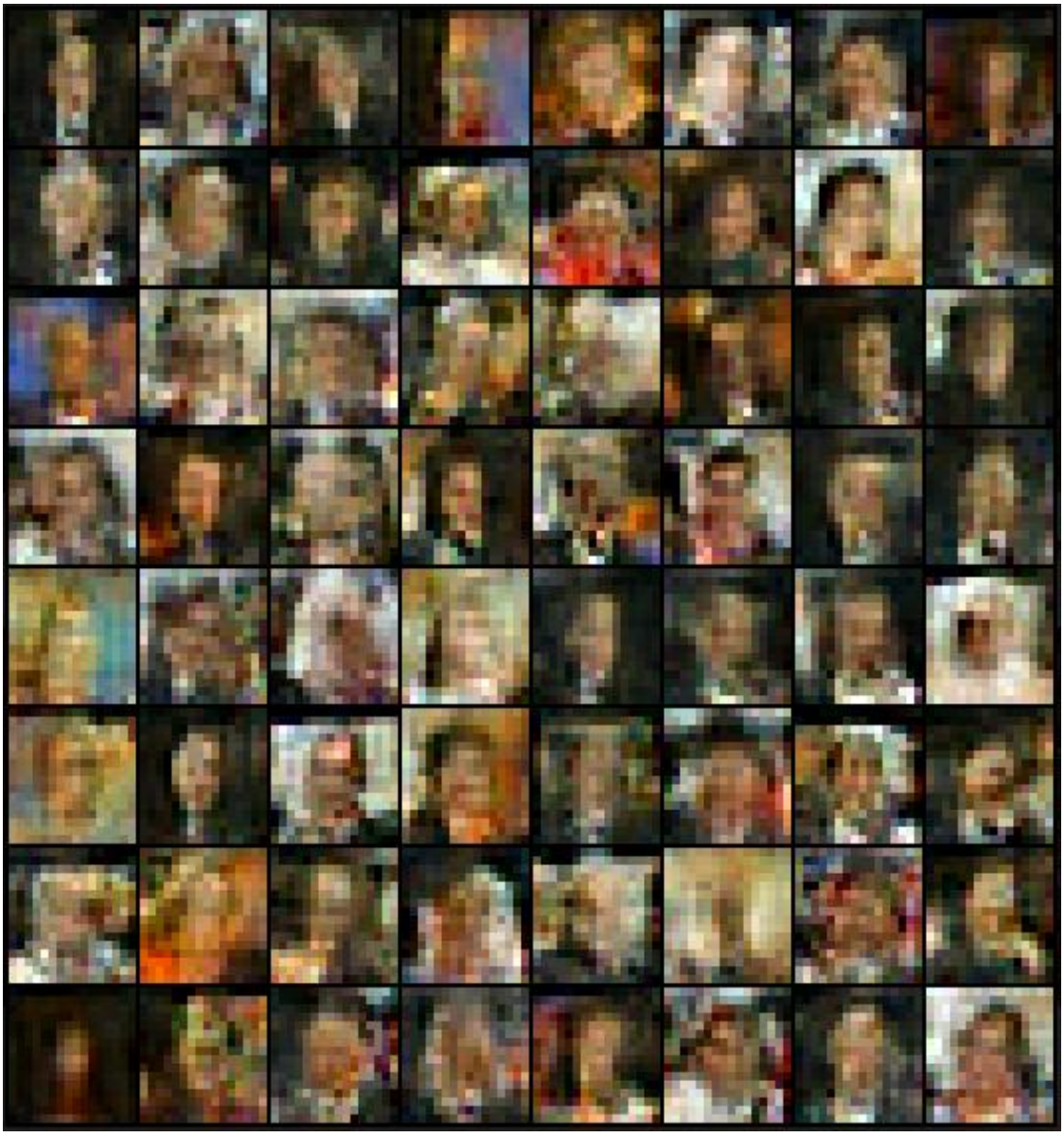


Figure 3.8: Generated Image during training process
[depth 3, epoch 120]

Figure 3.9: Generated Image during training process

[depth 4, epoch 120]

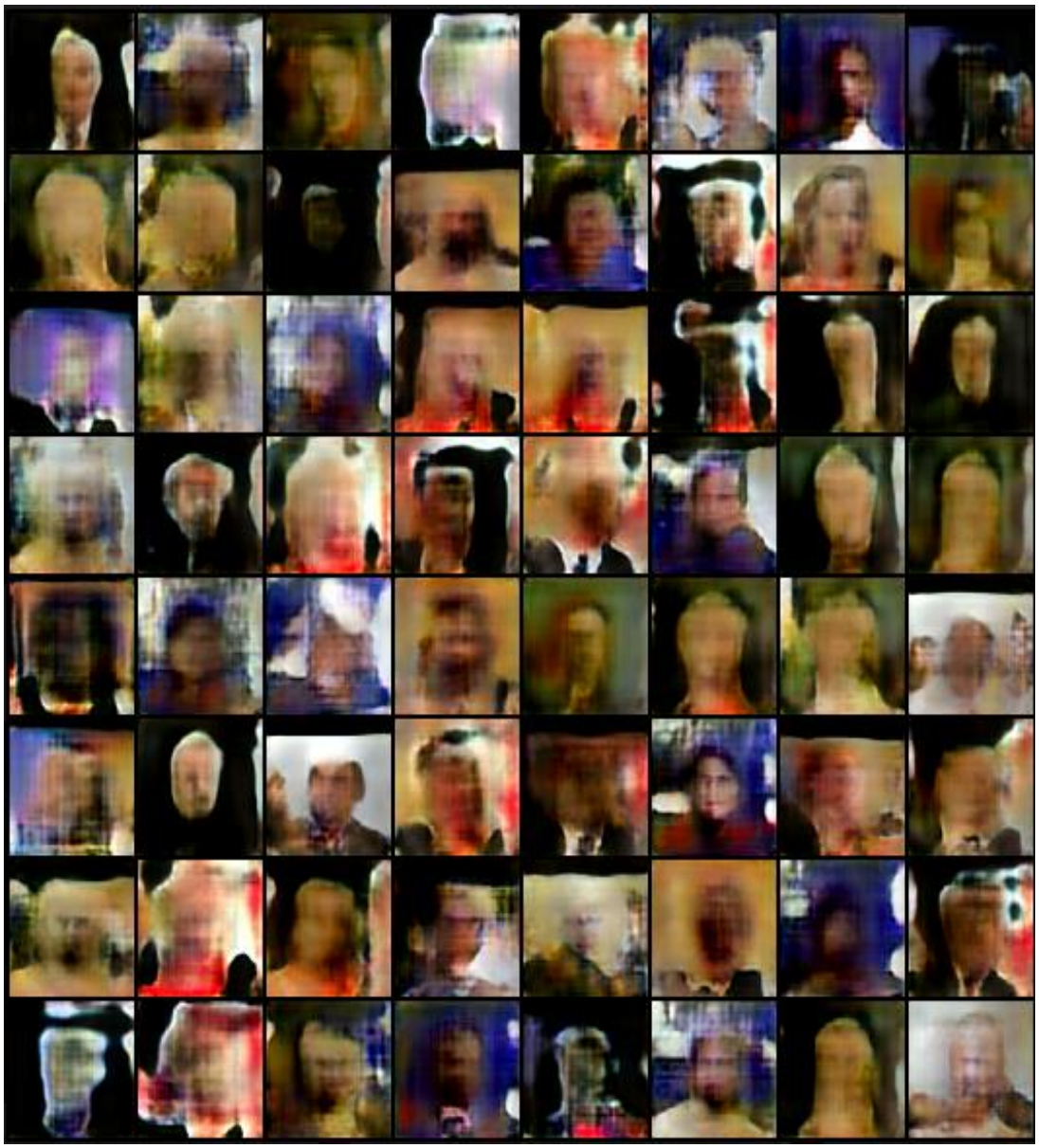


Figure 3.10: Generated Image during training process
[depth 5, epoch 120]

We also use two pretrained sentence encoder dataset:

- **Infersent**: [2] InferSent is a sentence embedding's method that provides semantic representations for English sentences. [2] It is trained on natural language inference data and generalizes well to many different tasks. In this project we used infersent2.pkl

- **GloVe**: [1] GloVe is an unsupervised learning algorithm for obtaining vector representations for words. [1] Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. We used 'glove.840B.300d.txt' file to train our model.

3.4.2 Generating Images from the User Description

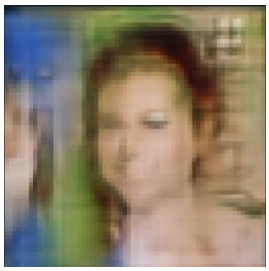
Code for generating output from user description is provided in Appendix C.3 .

3.4.3 Generated Output

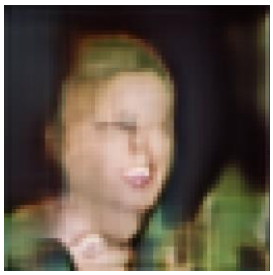
Here we have given 9-sample description to generate output and we got some amazing results but not the best in quality.

Our description contains lowercase letters only and a mandatory space after each comma/full stop and before comma/full stop. Moreover, we used space between two digit number (ex: we used 5 0 instead of 50).

Table 3.1: Sample input and output (9 output is generated)

Description	Output
a boy with blonde hair and smiling happy face	

a man in his late 5 0 s has an elongated face with a prominent nose . short mustache with a receding hairline and brown eyes	
an older adult male looking somewhat confused , angry or anxious . he seems to be from a mediterranean area .	
a man with a big smile. short blonde hair and with a very happy face .	
blonde hair happy woman	

long wavy blond hair , arched eyebrows , wide smile , bony cheek dimples .	
man with black hair	
man with smiling happy face and brown hair	
white male in his 4 0 ' s , pale complexion , huge bulging forehead probably as a result of hair loss , thin straight chocolate brown hair , dark eyes , arched eyebrows , thin pointed nose , thin upper lip , long face	

CHAPTER 4
EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Introduction

Generating face from a user's description is one of the most famous topics in GAN technology. So many researchers are working on GAN technology and inventing new GAN technologies day by day. GAN is more upgraded now. Text to image synthesis is a hot topic in the deep learning sector. Therefore, we want to experiment with our project model with different datasets and see the result how it performs. There are so many free datasets available on the internet, which contains so many photos of birds and flowers and celebrity photos. With which we can train our model and see the generated output if the generated output is better with them. We also want to try out other face generating or image synthesizing GAN models like MSG-GAN. We have heard that MSG-GAN performs better than ProGAN. Therefore, we want to perform some experiments on this model. There are also free datasets like CUB, COCO. The latest researches, which are applied for generating images from a piece of text of description of text, are mostly focusing on high-resolution photo generation, which will be photo realistic.

4.2 Experimental Results

We have considered the face descriptions of different humans over the world as experimental images. We have taken the images from Research in Vision and Language (Rival) group. We have considered celebrity faces with textual descriptions embedded with itself. It has been observed that the proposed GAN model output varies with the given textual description given by the user. The GAN model generates better output if the textual description is very descriptive. Simple description generates inaccurate images. We tried our model to generate the same image with the same description. But it failed to generate the same image with same input textual description.

4.3 Descriptive Analysis

In order to generate a real image, we had to train our model with depth 7 and dimension 256×256 but it will consume more resources, time, and space. That is why we reduced the depth level to 5 and the dimension to 64×64. As we use ProGAN architecture here, it generates images starting from lower resolution to higher resolution (4×4 → 8×8 → 16×16 → 32×32 → 64×64). In each depth, we have used 120 epochs with batch size 32 and fading percentage 50%. In our network, flow of data mainly depends on some points:

- In this process, we need textual descriptions but those are written in plain English. These textual descriptions are encoded in LSTM network is mainly a summary vector.
- In next step, that embedding has from LSTM passed through the CA (Conditional Augmentation) which is our single linear layer. From these step we can get textual part of the latent vector for our GAN model as input. Here this process uses VAE like re-parameterization technique.
- Another part of the latent vector is Gaussian noise. That is why the latent vectors produced in this GAN model is faded and in final layer of the discriminator the embedding also faded.
- New layer uses fade in technique to prohibit destroying previous learning.

4.4 Summary

The results of the study show that image generation from the textual descriptions using GAN technology is a very powerful method. Though the output is not highly accurate and progressive growing or ProGAN method is mainly an automated method, ProGAN is a very powerful technique, which can train our GAN model faster and in a stable manner.

CHAPTER 5
SUMMARY, CONCLUSION, RECOMMENDATION AND IMPLICATION FOR FUTURE RESEARCH

5.1 Summary of the Study

We have considered few limitations to the system. First drawback of this project is that we cannot generate high quality output image from the given description, so generating the output image was a challenge for us. Another drawback would be the model uses high GPU resources to generate the image. The other drawback can be the algorithm, which fades the background too much for image analysis of face generation. Because of the faded background of the picture, it might not show promising or accurate results. In the initial development of this project, we faced many challenges regarding this problem. Another problem we faced the resulted photo was stored in the directory but it was not shown on the terminal. For the time being, we allowed the user to give the description in the terminal and the user can see the output on the terminal screen. To show the image on the terminal we used matplotlib library package. We found another limitation; if we want to generate images with the same description repeatedly, it cannot remember the previously generated image with the same description. It will generate a new one and replace the old one.

5.2 Conclusions

Criminal activities are growing day by day and our eagerness to see a face of which we have read about in novels. This can be only done by text to face generation or text to image synthesis. Making a sketch of a criminal is a need of modern community for our own security. Beside we are really eager to know how our novels characters would look like with a real face. This project can fulfil both needs. In this project, we are training the text-encoder and image-decoder at the same time.

5.3 Recommendations

As shown in the output, our GAN model can generate low-resolution images based on the given description by the user. The output image becomes faded if we try higher resolution with higher depth level. The background of the image becomes noisy and the picture looks like a pixelated image. Which is not a very good outcome. There are other GAN models architecture, which are more efficient and accurate in terms of generating high-resolution output than ProGAN and StackGAN. FTGAN (Fully Trained Generative Adversarial Network) is one of them. FTGAN can generate photo-realistic images whose quality is close to the ground-truth. There are several drawbacks of using ProGAN and StackGAN. One of the drawbacks of using StackGAN is it can be a sort of overkill for any given distribution matching problem. On the other hand, ProGAN can be trained only once at a time which is progressively systematic increasingly refined or higher resolutions. However, we used the combined version of ProGAN and StackGAN but still the output is not up to mark. However, there is a solution to get better output from ProGAN. The details are provided in Appendix B.2.

Hence, the output is not good with our model architecture; we recommend using MSG-GAN or FT-GAN for text to face generation.

5.4 Implication for Further Study

We have planned to update this project for a new dataset. The dataset we used in this project is not enough to train our GAN model for generating picture perfect image. We want to use the Celeb-A dataset that contains 1000 images of random celebrities. It will definitely generate a real image from the user description. We will make the model train in a manner in which it will be able to generate the same face with the same description. It will not forget the previously generated data. It will be able to learn by itself each time a user will try to generate a face with our project. We will build an interface to give the description and generate the image in multiple level. So that the user can identify the right or desired face from the description as he expected (i.e identifying the suspect's face). We will try to use different GAN model for better accuracy of the generated image.

APPENDICES

Appendix A: Research Reflection

A.1

Progressively growing GAN is mainly an automated process where at the beginning it generates images with low resolution i.e. 4 x 4, and then it increases the size of resolution and add new layers with the previously generated images.

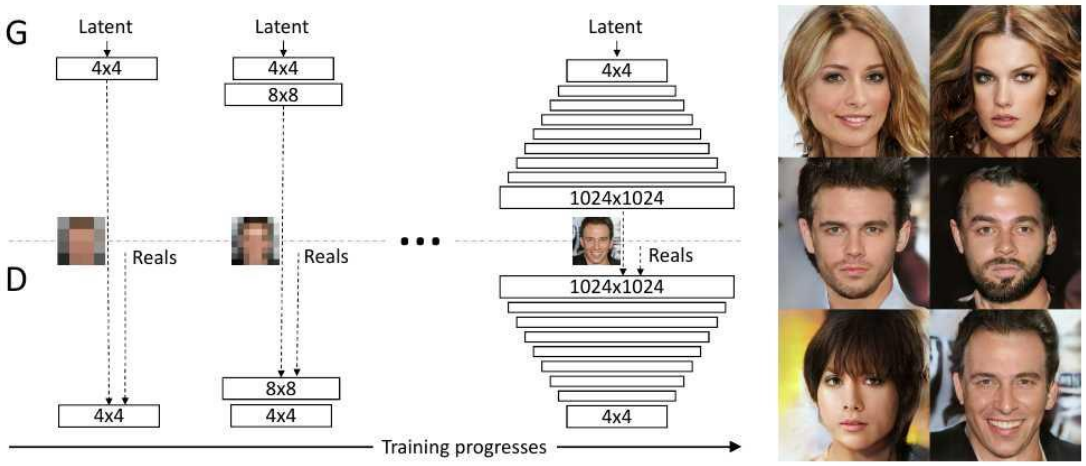


Figure A1: Main training process of ProGAN method. [8]

Appendix B: Related Issues

B.1

Google Colab is mainly based on jupyter notebook and provided by google. The extra feature is 'Google Colab' provides us hosted GPU (1xTesla K80, having 2496 CUDA cores, compute 3.7, 12GB (11.439GB Usable) GDDR5 VRAM).

B.2

We use batch size or batch normalization method in ProGAN architecture. However, the output is not so good. Because batch normalization mainly have trainable weights, which creates noises in image, but in this case, we can use Pixel Normalization method. This method does not have any trainable weights, so it normalizes the feature vectors according to pixel size.

Appendix C: Related Codes

C.1

Code for generating bar graph of loss data:

```
import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

for i in range(4, 5):

  log=pd.read_csv('/content/gdrive/My Drive/T2F/training_runs/2/losses/loss_{}.log'.format(i), delimiter='\t', header=None, names=['d_loss', 'g_loss', 'kl_loss'])

  n = len(log.d_loss)

  xs = list(range(n*i, n*(i + 1)))

  plt.bar(xs, log.d_loss, color='red')
 plt.bar(xs, log.g_loss, color='yellow')
 plt.bar(xs, log.kl_loss, color='green')

 plt.legend(log.columns)
```

C.2

Code for generating line plot diagram of loss data:

```
import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

for i in range(4, 5):

  log = pd.read_csv('/content/gdrive/My Drive/T2F/training_runs/2/losses/loss_{}.log'.format(i), delimiter='\t', header=None, names=['d_loss', 'g_loss', 'kl_loss'])

  n = len(log.d_loss)

  xs = list(range(n*i, n*(i + 1)))

 plt.plot(xs, log.d_loss)

 plt.plot(xs, log.g_loss)
```

```
plt.stem(xs, log.kl_loss)
plt.legend(log.columns)
```

C.3

Code for generating output from user description:

```
import torch as th
import numpy as np
import data_processing.DataLoader as dl
import yaml
current_depth = 5
from networks.TextEncoder import Encoder
from networks.ConditionAugmentation import ConditionAugmentor
from networks.C_PRO_GAN import ProGAN
# define the device for the training script
device = the.device("cuda" if th.cuda.is_available() else "cpu")
#load the generator.
def get_config(conf_file):
    """
    parse and load the provided configuration
    :param conf_file: configuration file
    :return: conf => parsed configuration
    """
    from easydict import EasyDict as edict
    with open(conf_file, "r") as file_descriptor:
        data = yaml.load(file_descriptor)
    # convert the data into an easyDictionary
    return edict(data)
```

```
config = get_config("configs/2_colab.conf"
c_pro_gan = ProGAN(
    embedding_size=config.hidden_size,
    depth=config.depth,
    latent_size=config.latent_size,
    learning_rate=config.learning_rate,
    beta_1=config.beta_1,
    beta_2=config.beta_2,
    eps=config.eps,
    drift=config.drift,
    n_critic=config.n_critic,
    device=device
)
c_pro_gan.gen.load_state_dict(th.load("training_runs/2/saved_models/GAN_GEN_4_
120.pth"))
#load my embedding and conditional augmentor
dataset = dl.Face2TextDataset(
        pro_pick_file=config.processed_text_file,
        img_dir=config.images_dir,
        img_transform=dl.get_transform(config.img_dims),
        captions_len=config.captions_length
    )
text_encoder = Encoder(
        embedding_size=config.embedding_size,
        vocab_size=dataset.vocab_size,
        hidden_size=config.hidden_size,
        num_layers=config.num_layers,
```

```
        device=device
    )
text_encoder.load_state_dict(th.load("training_runs/2/saved_models/Encoder_4_120.pth"))
condition_augmenter = Condition Augmentor(
        input_size=config.hidden_size,
        latent_size=config.ca_out_size,
        device=device
    )
condition_augmenter.load_state_dict(th.load("training_runs/2/saved_models/Condition_Augmentor_4_120.pth"))
# #ask for text description/caption
#caption to text encoding
caption = input('Enter your desired description : ')
seq = []
for word in caption.split():
    seq.append(dataset.rev_vocab[word])
for i in range(len(seq), 100):
    seq.append(0)
seq = th.LongTensor(seq)
seq = seq.cuda()
print(type(seq))
print('\nInput : ', caption)
list_seq = [seq for i in range(0,1)]
print(len(list_seq))
list_seq = th.stack(list_seq)
list_seq = list_seq.cuda()
embeddings = text_encoder(list_seq)
```

```
c_not_hats, mus, sigmas = condition_augmenter(embeddings)
z = th.randn(list_seq.shape[0],
             c_pro_gan.latent_size - c_not_hats.shape[-1]
            ).to(device)
gan_input = th.cat((c_not_hats, z), dim=-1)
#alpha=0.1
alpha = 0.007352941176470588
#alpha =0.00001
samples=c_pro_gan.gen(gan_input,
                      current_depth,
                      alpha)
from torchvision.utils import save_image
from torch.nn.functional import upsample
#from train_network import create_grid
img_file = caption + '.png'
samples = (samples / 2) + 0.5
if int(np.power(2, c_pro_gan.depth - current_depth - 1)) > 1:
    samples = upsample(samples, scale_factor=current_depth)
#save image to the disk, the resulting image is <caption>.png
save_image(samples, img_file, nrow=int(np.sqrt(20)))
# #output the image.
```

C.4

Required config file details for training our model:

```
# Hyperparameters for the Model
captions_length: 100
```

```
img_dims:
  - 64
  - 64
# whether to use a pretrained encoder:
use_pretrained_encoder: True
# LSTM hyperparameters
# embedding_size: 128
# hidden_size: 512
# num_layers: 3  # number of LSTM cells in the encoder network
# Conditioning Augmentation hyperparameters
hidden_size: 4096  # output size of the Pretrained encoder
ca_out_size: 128
compressed_latent_size: 32
# Pro GAN hyperparameters
use_eql: True
use_ema: True
ema_decay: 0.999
depth: 5
latent_size: 256
learning_rate: 0.001
beta_1: 0
beta_2: 0.99
eps: 0.00000001
drift: 0.001
n_critic: 1
# Training hyperparameters:
epochs:
```

```
  - 120
  - 120
  - 120
  - 120
  - 120
# % of epochs for fading in the new layer
fade_in_percentage:
  - 50
  - 50
  - 50
  - 50
  - 50
batch_sizes:
  - 64
  - 64
  - 64
  - 64
  - 32
loss_function: "wgan-gp"  # the loss function to be used
num_workers: 3
feedback_factor: 1  # number of logs generated per epoch
checkpoint_factor: 10 # save the models after these many epochs
use_matching_aware_discriminator: True  # use the matching aware discriminator
```

REFERENCES

[1]J. Pennington, "GloVe: Global Vectors for Word Representation", Nlp.stanford.edu, 2019. [Online]. Available: https://nlp.stanford.edu/projects/glove/. [Accessed: 31- Oct- 2019].

[2]A. Conneau, D. Kiela, H. Schwenk, L. Barrault and A. Bordes, "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data", arXiv.org, 2019. [Online]. Available: https://arxiv.org/abs/1705.02364. [Accessed: 01- Nov- 2019].

[3]H. Zhang et al., "StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks", arXiv.org, 2019. [Online]. Available: https://arxiv.org/abs/1710.10916. [Accessed: 01- Nov- 2019].

[4]A. Karnewar, "FAGAN: Full Attention GAN", Medium, 2019. [Online]. Available: https://medium.com/@animeshsk3/fagan-full-attention-gan-2a29227dc014. [Accessed: 01- Nov- 2019].

[5]S. Wolf, "ProGAN: How NVIDIA Generated Images of Unprecedented Quality", Medium, 2019. [Online]. Available: https://towardsdatascience.com/progan-how-nvidia-generated-images-of-unprecedented-quality-51c98ec2cbd2. [Accessed: 01- Nov- 2019].

[6]R. Horev, "Explained: A Style-Based Generator Architecture for GANs - Generating and Tuning Realistic Artificial Faces", Medium, 2019. [Online]. Available: https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431. [Accessed: 01- Nov- 2019].

[7]Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments", Vis-cs.umass.edu, 2019. [Online]. Available: http://vis-www.cs.umass.edu/lfw/lfw.pdf. [Accessed: 01- Nov- 2019].

[8]"GANs for Image Generation: ProGAN, SAGAN, BigGAN, StyleGAN - CV Notes", Cvnote.ddlee.cn, 2019. [Online]. Available: https://cvnote.ddlee.cn/2019/09/15/ProGAN-SAGAN-BigGAN-StyleGAN.html. [Accessed: 01- Nov- 2019].

Himu_Text to Face

ORIGINALITY REPORT

14%	7%	4%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	4%
2	medium.com Internet Source	1%
3	Submitted to University of Westminster Student Paper	1%
4	iamaaditya.github.io Internet Source	1%
5	ajouronline.com Internet Source	1%
6	Submitted to University of Leeds Student Paper	1%
7	www.iloencyclopaedia.org Internet Source	1%
8	Submitted to Kyungpook National University Student Paper	1%
9	Submitted to City University of Hong Kong Student Paper	<1%

10	en.m.wikipedia.org Internet Source	<1%
11	pythondata.com Internet Source	<1%
12	Changliang Li, Yixin Su, Wenju Liu. "Text-To-Text Generative Adversarial Networks", 2018 International Joint Conference on Neural Networks (IJCNN), 2018 Publication	<1%
13	Submitted to AUT University Student Paper	<1%
14	Submitted to University of Hong Kong Student Paper	<1%
15	export.arxiv.org Internet Source	<1%
16	Submitted to University of Queensland Student Paper	<1%
17	Abhishek Verma, Vanshika Mittal, Suma Dawn. "FIND: Fake Information and News Detections using Deep Learning", 2019 Twelfth International Conference on Contemporary Computing (IC3), 2019 Publication	<1%
18	aclweb.org Internet Source	<1%