FGTD: Face generation from textual description

Kalpana Deorukhkar[1], *Kevlyn Kadamala[2] and Elita Menezes[3]

[1]kalpanas@fragnel.edu.in, [2]kevlyn@gmail.com,
[3]elitae.menezes@gmail.com

[1,2,3] Father Conceicao Rodrigues College of Engineering, Mumbai, Maharashtra-400050, India
* Corresponding Author

Abstract. Majority of current text-to-image generation tasks are limited to creating images like flowers (Oxford 102 Flower), birds (CUB-200-2011), and Common Objects (COCO) from captions. The existing face datasets such as Labeled Faces in the Wild and MegaFace lack description while datasets like CelebA have attributes associated but do not provide feature descriptions. Thus, in this paper we build upon an existing algorithm to create captions with the attributes provided in the CelebA dataset, which can not only generate one caption but it can also be extended to generate N captions per image. We utilise Sentence BERT to encode these descriptions into sentence embeddings. We then perform a comparative study of three models - DCGAN, SAGAN and DFGAN, by using these sentence embeddings along with a latent noise as the inputs to the different architectures. Finally, we calculate the Inception Scores and the FID values to compare the output images across different architectures.

Keywords: Generative adversarial networks, Face generation, Text to image generation, Caption creation, Natural language processing, Deep learning, Text to face generation

1 Introduction

In the field of Generative Adversarial Networks, there has been a lot of advancement since its inception in 2014 [1]. In 2016, we saw a new type of GAN architecture that could generate images from text [2]. This text-to-image problem can be considered as the reverse of a text captioning problem. Similar to text caption, text-to-image helps to understand the semantic relationship between text and image. Text-to-face synthesis is a text-to-image subdomain, aiming to synthesize face images based on human descriptions. For example, it involves describing facial features like “Black Hair” and “Oval Face” to the RGB pixel space as stated in [8]. While there has been a lot of advances in the text-to-image domain [3, 27, 29, 30, 7], the same coverage is not shared in the domain of text-to-face [8, 9].

Datasets in the text-to-image domain include the Oxford 102 Flower [12] which contains semantic descriptions of flowers and the CUB-200-2011 [13] which contain descriptions of birds while the Common Objects (COCO) [14] broadly describes the object and its context. However, none of these datasets contains any physical description of faces that are required for generating faces. While extensive research has already been conducted to create facial datasets the existing face datasets such as Labeled Faces in the Wild [10] and MegaFace [11] lack description. While the CelebA [15] dataset has a large number of images they have a list of 40 attributes associated with it. These attributes cannot be directly used as they would produce poor-quality semantic vectors. Therefore, we opted in for creating our algorithm for the dataset. Using these attributes, we generated random but meaningful sentences while avoiding grammatical errors.

The method of text-to-face follows a similar procedure when compared to text-to-image. It requires an encoder that encodes the sentences to convert them to a semantic vector. This semantic vector is then fed to the GAN with a noise vector to conditionally generate images.

While working with this methodology, it is important to generate high-quality semantic vectors. Existing solutions to generate sentence embeddings include SkipThought [16] which trains an encoder-decoder architecture to predict the sentences around it. The Universal Language Model [17] can be fined tuned on an existing dataset from which embeddings can be obtained. Sentence BERT [18], a modification of the pre-trained BERT [34] network can also be used for the same task. These models can be used to capture the facial features while maintaining semantic consistency.

Once, the text has been encoded, the next stage includes the generation of images. The GAN models we have studied and evaluated in this paper include the following architectures: DCGAN [19], DFGAN [3], Self-Attention GAN [20]. Keeping resource restrictions in mind, these models have successfully facilitated face generation from textual descriptions. While performing this task we need to ensure that the images generated match their textual description. For this purpose, the discriminator not only has to distinguish whether the images are real or fake but also determine whether the given image-text pair match, as given in [2]. To evaluate the performance of GANs we used three scoring metrics: (1) Inception Score [22], (2) Fréchet Inception Distance (FID) score [23] and Clean FID [4]. We then tabulated these results which could be used as a reference for other research purposes.

Fig. 1. Comparison of the generated images from the DCGAN, SAGAN, DFGAN for both single captions as well as N captions along with its corresponding ground truth image.

Hence, our aim in this paper can be divided into three sections. (1) To create a dataset of facial images with rich textual descriptions. (2) To compare the different text-to-image architectures. (3) To log and evaluate these models using Weights and Biases [21].

2 Related Work

There has been significant research done in the field of Generative Adversarial Networks. Research has been conducted in the fields of audio [24], video [25], and text [26]. However, in this section, we shall mainly focus on two domains: (1) Text-to-image (2) Text-to-face.

2.1 Text-to-Image:

Text-to-image synthesis was a novel concept when it was introduced by Scott Reed et al. [2] in 2016. They made use of DCGAN [19] which was conditioned on text features encoded by a hybrid character-level convolutional recurrent neural network. Since then, this field has seen significant progress in generating high-quality images which also maintain consistency with their descriptions. Han Zhang et al. introduced a method that adopts the method of stacking multiple generators and discriminators [27]. Later they introduced StackGAN-v2 which offered a more stable training behavior than StackGAN-v1 by jointly approximating multiple distributions [28].

With an improvement in the quality of images, the focus was now shifted to improving the similarity between the output images and the input descriptions. AttnGAN introduced the concept of an attentional generative network. By paying attention to the related terms in the natural language description, they were able to synthesize fine-grained details at different subregions of the image.[29]. The concept of attention is also seen in [20] along with spectral normalization [40]. They reported that the self-attention module is effective in modeling long-range dependencies while spectral normalization helped stabilize GAN training. Meanwhile, Zizhao Zhang et al. proposed hierarchical-nested adversarial objectives inside the network. This methodology computes the matching-aware pair loss and the local image loss at different image resolutions [30]. In [3], the authors proposed a simpler and more efficient method to generate realistic and text-matching images. It generates high-resolution images by using a target aware discriminator to generate high-quality images and at the same time maintain a text-image consistency without introducing an extra network.

Table 1. Comparison of Inception Score between the different Text-to-Image Architectures on the CUB, Oxford and COCO datasets

Models	CUB	Oxford	COCO
GAN-INT-CLS	2.88 ± .04	2.66 ± .03	7.88 ± .07
StackGAN	3.70 ± .04	3.20 ± .01	8.45 ± .03
AttnGAN	4.36 ± .03	-	25.89 ± .47
HDGAN	4.15 ± .05	3.45 ± .07	11.86 ± .18
DFGAN	5.10	-	-

2.2 Text-to-Face

Text-to-face synthesis follows a procedure similar to that of text-to-image, however, the research done in this domain is quite limited. Datasets such as Labeled Faces in the Wild [10], MegaFace [11], and CelebA [15] lack the textual descriptions that are associated with the images. Nevertheless, the CelebA dataset has about 200k images of human faces and provides different attributes for different faces. There have been approaches to convert these attributes to meaningful descriptions [8], however, we found that the caption creation process lacks variation and clarity. Meanwhile, there exists a crowdsourced dataset called Face2Text [33] with 400 images and their descriptions, however, it is not freely available to the general public.

Some of the initial research done in this domain was conducted by Xiang Chen et al. where they introduced FTGAN. Here they proposed to train the text encoder and the image decoder at the same time. Although this architecture performed well and produced high-quality images, they were unstable during training [9]. Meanwhile, the author of [31] made use of an LSTM network to encode the textual descriptions into a summary vector. This embedding vector is fed as an input to the generator, while for the discriminator, it is fed to its final layer. The training procedure for this GAN is similar to the ProGAN [32] paper where it increases its spatial resolutions, layer by layer. In [8], the authors used the DCGAN [19] architecture with a matching aware discriminator [2]. They also made use of SkipThought Vectors [16] to encode text into embeddings. While their model did not face mode collapse, they only generated images of 64x64 resolution, which is quite low compared to other work [9, 31].

As a result, to provide a clear and descriptive caption we decided to create our own algorithm. We also work on the ideas presented previously to train and evaluate different architectures to produce a table of results.

3 Background

In this section, we provide an overview of caption creation, sentence vectors, GANs, and their different architectures that we have built upon. Finally, we talk about the three different scoring metrics that we use for evaluation.

3.1 Caption Creation

In [9] the authors built a dataset called the SCU-Text2Face. This dataset is based on CelebA [15] and contains 1000 images. Here for each image, there are five descriptions, however, these descriptions were given by different people. On the other hand, [8] proposed to build an algorithm to generate captions based on the attributes given by CelebA. This involved creating six groups of facial characteristics in response to six questions that describe the face in a step-by-step manner, beginning with the facial outline and ending with the facial features that decide its appearance.

3.2 Sentence Vectors

As input to the generator, we need to provide a semantic vector of the sentence. To achieve this, we used Sentence BERT [18]. Sentence BERT fine-tunes BERT in a Siamese/triple network architecture. It is a modification of BERT [34] to derive semantically meaningful sentencing embeddings. The authors showed that the different methods for obtaining sentence embeddings using BERT gave poor results on tasks like textual similarity. They also compared the computational efficiency of SBERT to GloVe embeddings [37], InferSent [36] and Universal Sentence Encoder [35]. Compared to InferSent and Universal Sentence Encoder it is 9% and 55 % faster respectively. This is due to the smart batching strategy that is used in which sentences with similar lengths are grouped together. They are then padded to the longest element in the mini-batch, reducing the overhead in computing the padding tokens.

3.3 Generative Adversarial Networks

In GANs, we train two models simultaneously: The Generator (G) is responsible for capturing the data distribution and the Discriminator (D) is responsible for identifying if the sample came from the training data or Generator. This framework adopts the minimax two-player game strategy. The training procedure for the G is to maximize the probability of the D making a mistake and for D is to maximize the probability of assigning the correct label to both the training examples and examples generated from the generator [1].

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))] \quad (1)$$

3.4 Model Overview

DCGAN. The architecture comprises of strided convolutions in the discriminator and fractional-strided convolutions in the Discriminator. In the Discriminator, it uses LeakyReLU [39] as an activation function for all layers. In the Generator, it uses ReLU [38] as an activation function in all layers except for the output layer. DCGAN is a more stable set of architecture for training Generative Adversarial Networks and the authors provide substantial evidence for the same. However, it does suffer from some forms of model instability like mode collapse [19].

DFGAN. The architecture of the proposed DF-GAN comprises a generator, discriminator, and a pre-trained text encoder. DF-GAN generates images with a high resolution directly by one pair of generator and discriminator and combines the text information and visual feature maps through multiple Deep text-image Fusion Blocks (DFBlock) in UPBlocks. Armed with Matching-Aware Gradient Penalty (MA-GP) and one-way output, the model can generate more realistic and text-matching images [3].

SAGAN. In [20], the authors introduced a self-attention mechanism into convolutional GANs. They introduced this mechanism into the generator as well as the discriminator. While AttnGAN [29] used attention over word embeddings with an input sequence, it did not apply self-attention over internal model states. SAGAN, however, learns to efficiently find global, long-range dependencies within internal representations of images. In order to stabilize training, they also proposed the use of spectral normalization [40]. This technique imposes global regularization and is also computationally light.

3.5 Evaluation and Scoring

In order to evaluate the images generated, Inception Score [22] and Fréchet Inception Distances [23] are used. IS uses the Inception model [6] to calculate the KL divergence between the conditional distribution and the marginal distribution. A higher Inception Score indicates that higher quality of images has been generated as well as each image is part of a particular class indicating higher diversity. However, as mentioned in [8], the images in CelebA[15] have high intraclass similarity due to similar facial features being present thus, making inception score a poor choice for evaluation. Hence, as an additional metric for evaluation, we included FID which computes the Fréchet distance that is used to calculate the distance between the feature vectors of real and generated images. Lower scores, in this case, indicate that the images generated are more realistic. We also noted that the authors in [4] found that FID calculation involves steps that produce inconsistencies in the final metric. As a result, we included Clean FID scores as our third evaluation metric.

4 Methodology

4.1 Process Flow

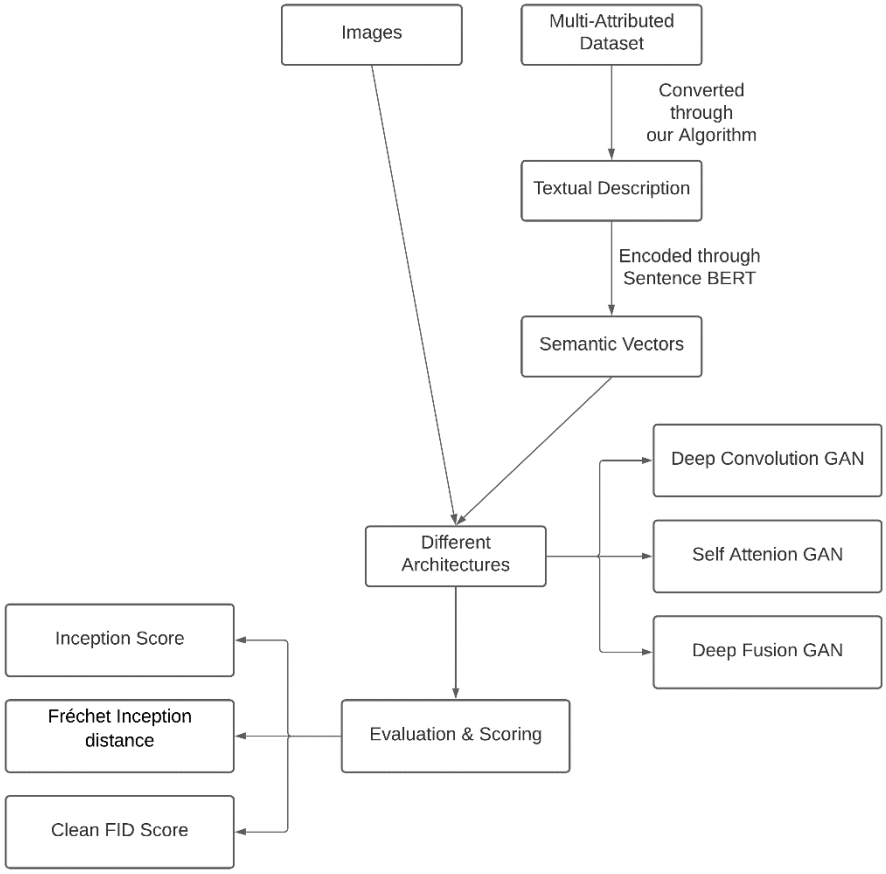

Fig. 2. Process flow for the training methodology

The process flow for our training methodology is given in Fig 2. The first step is creating the dataset. The corresponding attributes of each image are converted into a textual description. The final dataset contains the image id from CelebA [15] along with its corresponding description. Each batch of the dataset contains batches of images with its respective text, providing us with an image-text pair. The training steps are as follows:

1. The text from the image-text pair is encoded into a semantic vector with the help of Sentence BERT [18].

2. These semantic vectors are then used as the inputs to the different generator and discriminator models.
3. After each epoch, the losses are monitored with the help of Weights and Biases [21].
4. Once the training completes, each architecture is evaluated with the Inception Score [22], FID values [23], and Clean FID values [4].

4.2 Caption Creation

Similar to the authors in [8], we have divided the attributes present in CelebA [15] into six categories (see Table 2). This helps in segregating facial features into similar categories which makes it easy to add variation in their descriptions. We designed our algorithm with the aim to create 'N' captions using the given list of attributes. We achieved this by randomly choosing elements from a given list of options. As a result, we were able to achieve several variations of a certain phrase which enables the network to learn a wider variety of sentences.

Table 2. Attribute Categories for Caption Creation

Category	Attributes
Face Structure	Chubby, Double Chin, Oval Face, High Cheekbones
Facial Hair	5 o'Clock Shadow, Goatee, Mustache, Sideburns
Hairstyle	Bald, Straight Hair, Wavy Hair, Black Hair, Blond Hair, Brown Hair, Gray Hair, Receding Hairline
Facial Features	Big Lips, Big Nose, Pointy Nose, Narrow Eyes, Arched Eyebrows, Bushy Eyebrows, Mouth Slightly Open
Appearance	Young, Attractive, Smiling, Pale Skin, Heavy Makeup, Rosy Cheeks
Accessories	Wearing Earrings, Wearing Hat, Wearing Lipstick, Wearing Necklace, Wearing Necktie, Eyeglasses

Each category in our algorithm has its own function. The input to each function are the corresponding attributes to the category. The function then outputs a sentence that describes these attributes in a meaningful way. Each function has its own base case and scenarios, and to introduce variations, different choices of words are used. However, while doing so we ensured that there is no compromise in the grammatical structure. When a particular category has multiple attributes associated with it, sentence structure is randomly chosen and in cases of binary attributes, an equal probability is given.

Algorithm 1 Caption Generation for Facial Structure

faceAttributes # List of attributes present in facial structure category
isMale # Whether the face is of a male
function GenerateFacialStructure(*faceAttributes, isMale)*

features = {Chubby: […], *HighCheekbones:* [...], *OvalFace:* […], *DoubleChin:* […]}
if *isMale* **then**
sentence = Pick a random sentence with a male gendered noun
else
sentence = Pick a random sentence with a female gendered noun
if len(*faceAttributes)* = 1 **then**
attribute = *faceAttributes*[0]
sentence = sentence + random(features[attribute])
else
for *attribute* ∈ *faceAttributes* **do**
sentence = sentence + random(features[attribute])
return *sentence* # Output sentence for this category

Before each function is executed, we segregate and store each attribute of the image into its respective category list. This list is then passed as inputs to the function and the output descriptions are appended into a string. The final string that describes the image contains the accumulated output of all the category functions. The variation helps in generating N captions per image, thus providing us with different phrases for the same image in each loop.

To ensure equal distribution of attributes in each training batch we balance the dataset as follows:

- First, we count the number of times each attribute was present.
- Then, using the counts we calculated the attribute weights.
- Finally, for each attribute associated with an image, we calculate the sum of the attribute weight. We use this as our image weights.

4.3 Sentence Encoding

In order to convert text into embeddings, SBERT [18] is used. Each sentence from the description is passed to the model and their embeddings are stored in a list. After each sentence is parsed, the embedding list is then averaged and reshaped to (1, 768). For batches of images, the (1, 768) output is stacked to form a batch of embeddings with the shape (|B|, 768) where B is a batch set.

4.4 Network Architectures

In this paper, we have trained three models: (1) DCGAN [19], (2) SAGAN [20], and (3) DFGAN [3]. Once the embeddings have been obtained, they are passed as the inputs to the generators of these models along with a noise vector of dimension 100.

DCGAN. The generator of the DCGAN encodes the text embeddings with the help of a fully connected layer. The outputs from the fully connected layer are then concatenated with the noise vector and reshaped into a vector of shape (|B|, 356, 4, 4). The model architectures are similar to the architectures mentioned in [8]. The concatenated vector is then passed through a set of transposed convolutional layers that allows this output vector to be upsampled into an image of size 128x128. The discriminator consists of convolutional layers that are responsible for downsampling the image. The text embeddings are passed to a fully connected layer, expanded and then concatenated with the outputs of the second layer. This concatenated output is then passed to the final layer of the discriminator, producing outputs ranging from 0 to 1. The learning rates for the generator and the discriminator are 0.0002 and 0.0001 respectively. The Adam optimizer [42] for the generator as well as the discriminator is set with $\beta_1 = 0.5$ and $\beta_2 = 0.5$.

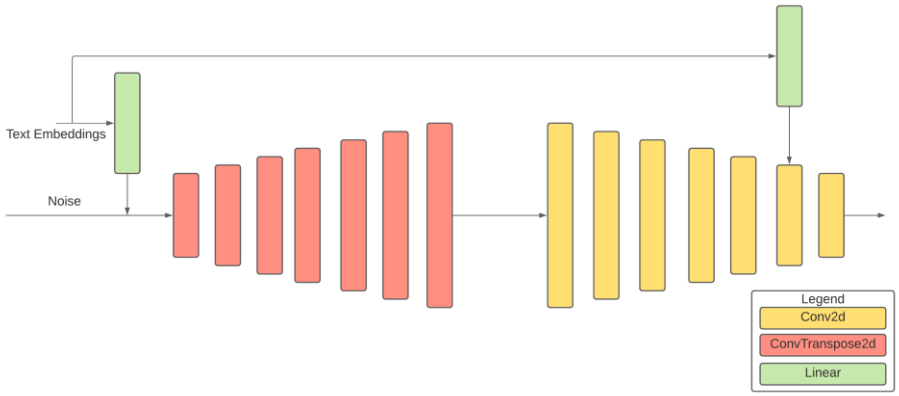

Fig. 3. Overview of Deep Convolution GAN which produces output images of size 128x128

SAGAN. The SAGAN has a slightly different architecture as compared to the architecture in [20]. Here, the generator has two fully connected layers with ReLU activations. The text embeddings pass through these layers reducing from vectors of length 768 to 256 and then finally to a vector of length 100 as the output of the final fully connected layer. This vector is then multiplied to the input noise and reshaped to a vector of size (|B|, 100, 1, 1). It passes through layers as stated in the SAGAN paper. For the discriminator, before the second last layer, the text embeddings are passed to a fully connected layer followed by a ReLU activation layer. This is concatenated to the output vectors of the previous layers. The resulting vector is then passed to two convolutional layers after which the output of the final convolutional layer passes through the sigmoid layer, generating outputs ranging from 0 to 1. Images of size 128x128 are generated. For the SAGAN, the learning rates of the generator and discriminator are 0.0001 and 0.0004 respectively. For both the models, the Adam optimizer [42] with $\beta_1 = 0$ and $\beta_2 = 0.9$ is used.

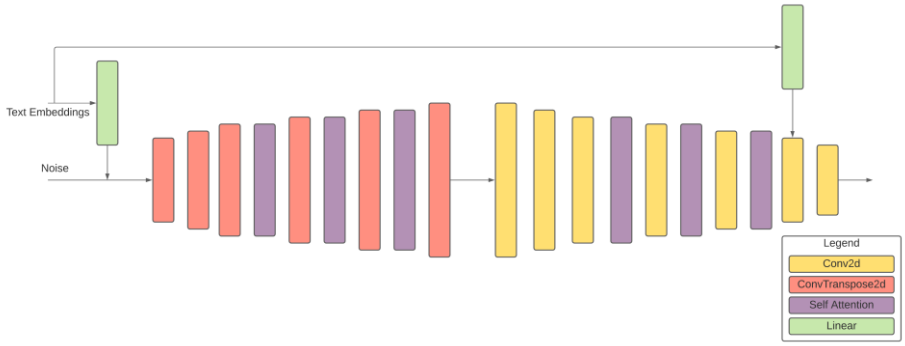

Fig. 4. Overview of Self Attention GAN which produces output images of size 128x128

DFGAN. The architecture of the DFGAN that we have used for training is very similar to the one used by the authors in [3]. However, the images generated are of size 128x128. In order to accommodate this reduced image size, the last block of the generator and the discriminator responsible for the generation and validation of 256x256 sized images have been omitted. A matching aware gradient policy was added to the discriminator which helped in improving the quality of the final image. For the DFGAN, the Adam optimizer [42] is set to $\beta_1 = 0$ and $\beta_2 = 0.9$ and the learning rates for the generator and discriminator are 0.0001 and 0.0004 respectively.

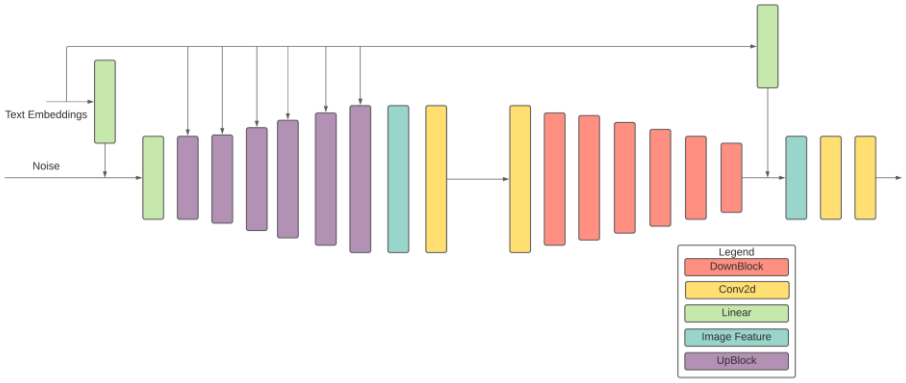

Fig. 5. Overview of Deep Fusion GAN which produces output images of size 128x128

5 Evaluation and Results

In order to calculate the inception score and FID values, we made use of tools provided in [5]. This library provides a fast and reliable evaluation of GANs in PyTorch. In addition to this, the authors in [4] provided a tool to calculate FID scores which removed

the inconsistencies present in earlier FID calculations [23]. Hence, as an additional metric, we have included Clean FID scores in the evaluation table (see table 2).

Table 3. Performance comparison between the different architectures on 1 caption and 5 captions datasets

	GAN	Inception Score	Fréchet Inception Distance	Clean FID
1 Caption	DCGAN	2.840 ± 0.062	87.146	87.580
	SAGAN	2.342 ± 0.029	114.512	115.256
	DFGAN	2.865 ± 0.041	109.140	106.453
5 Caption	DCGAN	2.732 ± 0.055	90.268	90.331
	SAGAN	2.855 ± 0.054	95.052	95.656
	DFGAN	3.455 ± 0.075	88.748	87.462

We initially trained our models on 10k images. However, it was observed that DCGAN suffered from mode collapse and could not recover over 100 epochs. We then increased the dataset size to 20k images. This resulted in stable image generation for all the models. The DCGAN and SAGAN models were trained for 20 epochs, while DFGAN was trained for 15 epochs. The loss patterns for the models are shown in Fig 6. These values were logged during training with the help of weights and biases [21]. This provided us with a dashboard that helped in model versioning and evaluation.

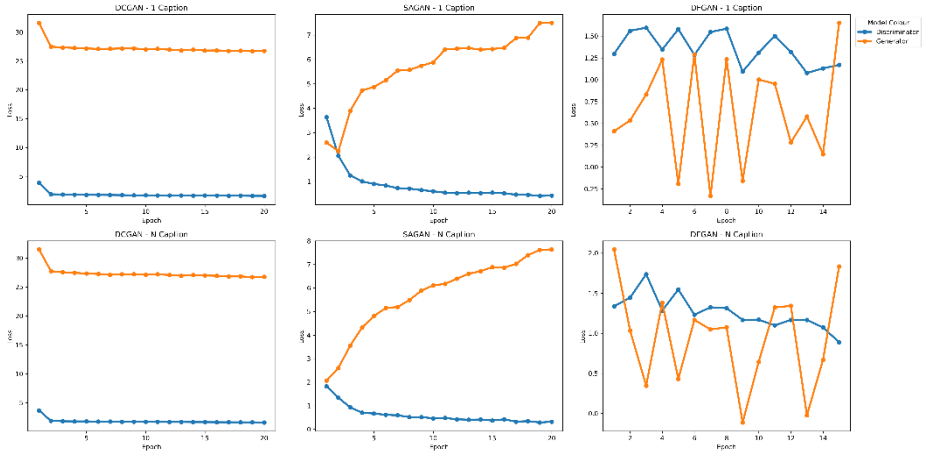

Fig. 6. Loss vs Epoch values comparison for the different architectures

The output images (Fig 7) show a close resemblance to their respective captions. For example, in the first row characteristics like high cheekbones, young and smiling can be identified. In row 2 features like bald, oval face and wearing sunglasses are clearly visible in the images. Attributes like smiling with a slightly open face are identifiable

in rows 3 and 5. The architectures have also been able to generate other accessories like a necktie which is pretty evident in the images on row 4.

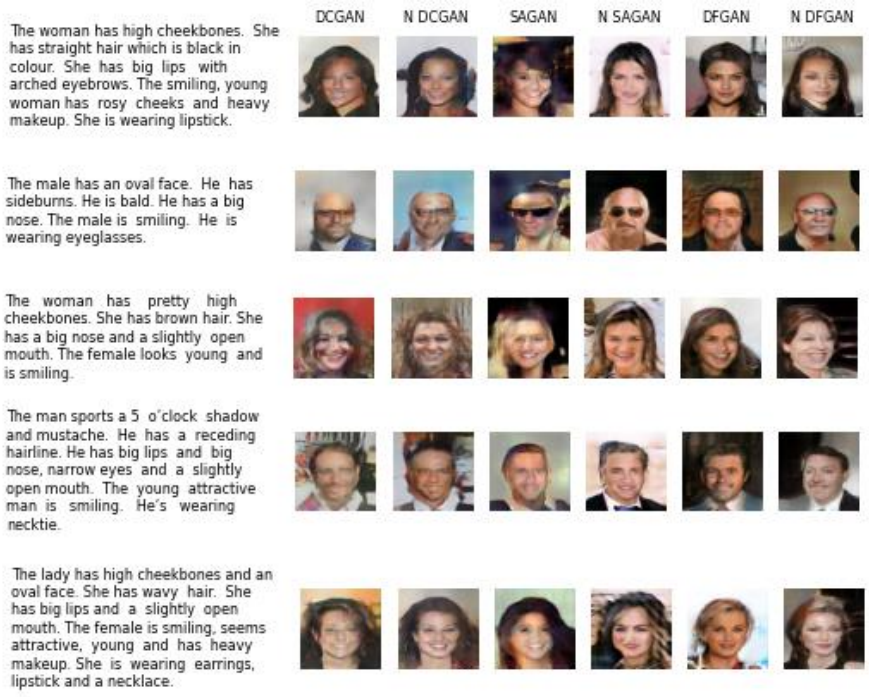

Fig. 7. Faces generated by the different architectures from a given text prompt

6 Conclusion and Future Work

In our work, we presented an algorithm to generate captions based on the CelebA dataset. This algorithm provides grammatically correct and meaningful sentences. We used these captions and tested different architectures of Generative Adversarial Networks to dive deeper into the lesser addressed domain of Text-to-Face. Due to the imbalanced distribution of the images in the CelebA dataset, images that contain less than 5 attributes or more than 12 attributes in their descriptions are under-represented. As a result, the images generated for these descriptions do not share the same quality when compared to the other images.

We found that by using SBERT, we were able to generate high-quality semantic vectors compared to the previous attempts using skip-thought vectors. From the three architectures, we identified that Self-Attention GANs and DFGANs produced higher quality images, however, due to the complex nature of their model architectures, they take more time to train when compared to the DCGAN. We also found that, for a given input, the DCGAN would generate batches of similar images, while the DFGAN and SAGAN would generate a wider variety of images while being semantically correct.

We believe it is due to the concatenation strategy of the noise with the output of the fully connected layer.

We then evaluated the images generated by the models with the help of three metrics, Inception Score, FID and the newly published Clean FID scores. Finally, we tracked our training runs with the help of Weights and Biases as a step towards reproducibility.

We believe that this work can be further improved by:

1. Introducing a better dataset balancing strategy that considers very short and extremely long descriptions.
2. Increasing the training steps for these models.
3. Extending the resolution of images to 256x256, 512x512 or further.
4. Using a transformer based model like DALL-E [41].

Ethics. We recognize how important of a role ethics plays in the development of AI. Datasets like CelebA have an unbalanced distribution of attributes, potentially leading to bias. For this reason, we have balanced the dataset to ensure that all attributes are well represented to eliminate bias. We also recognize that certain attributes like "attractive" are subjective. However, we are considering these labels as features for our model and not as a beauty standard.

Acknowledgement. We would like to thank our mentor Prof. Kalpana Deorukhkar for her constant support and guidance throughout the project.

References

1. Goodfellow, I.J, Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville A., Bengio, Y.: Generative Adversarial Networks. arXiv preprint arXiv:1406.2661 (2014)
2. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative Adversarial Text to Image Synthesis. In: International Conference on Machine Learning, pp. 1060-1069. PMLR (2016)
3. Tao, M., Tang, H., Wu, S., Sebe, N., Jing, X.Y., Wu, F., Bao, B.: Df-gan: Deep Fusion Generative Adversarial Networks for Text-to-Image Synthesis. arXiv preprint arXiv:2008.05865. (2020)
4. Parmar, G., Zhang, R., Zhu, J. Y.: On Buggy Resizing Libraries and Surprising Subtleties in FID Calculation. arXiv preprint arXiv:2104.11222 (2021)
5. Obukhov, A., Mseitzer, Willylulu, Zhydenko, S., Kyl, J., Lin, E.Y.-J.: High-fidelity Performance Metrics for Generative Models in PyTorch. https://github.com/toshas/torch-fidelity, Last accessed 1 March 2021
6. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826 (2016)
7. Hinz, T., Heinrich, S., Wermter, S.: Semantic Object Accuracy for Generative Text-to-Image Synthesis. arXiv preprint arXiv:1910.13321 (2019)

8. Nasir, O.R., Jha, S.K., Grover, M.S., Yu, Y., Kumar, A., Shah, R.R.: Text2FaceGAN: Face Generation from Fine Grained Textual Descriptions. In: 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), pp. 58-67. IEEE (2019)
9. Chen, X., Qing, L., He, X., Luo, X., Xu, Y.: FTGAN: A Fully-trained Generative Adversarial Networks for Text to Face Generation. arXiv preprint arXiv:1904.05729 (2019)
10. Huang, G., Mattar, M., Lee, H., Learned-Miller, E.G.: Learning to Align from Scratch. In: Advances in neural information processing systems, pp. 764-772 (2012)
11. Kemelmacher-Shlizerman, I., Seitz, S.M., Miller, D. and Brossard, E.: The Megaface Benchmark: 1 Million Faces for Recognition at Scale. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4873-4882 (2016)
12. Nilsback, M.E. and Zisserman, A.: Automated Flower Classification over a Large Number of Classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, pp. 722-729. IEEE (2008)
13. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 dataset (2011)
14. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L..: Microsoft coco: Common Objects in Context. In: European conference on computer vision, pp. 740-755 (2014)
15. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep Learning Face Attributes in the Wild. In: Proceedings of the IEEE international conference on computer vision, pp. 3730-3738 (2015)
16. Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., Fidler, S.: Skip-thought Vectors. arXiv preprint arXiv:1506.06726 (2015)
17. Howard, J., Ruder, S.: Universal Language Model Fine-tuning for Text Classification. arXiv preprint arXiv:1801.06146 (2018)
18. Reimers, N., & Gurevych, I.: Sentence-bert: Sentence Embeddings using Siamese Bert-networks. arXiv preprint arXiv:1908.10084 (2019)
19. Radford, A., Metz, L. and Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
20. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention Generative Adversarial Networks. In: International conference on machine learning, pp. 7354-7363. PMLR (2019)
21. Biewald, L.: Experiment Tracking with Weights and Biases, https://www.wandb.com/, (2020)
22. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.:Improved Techniques for Training Gans. arXiv preprint arXiv:1606.03498 (2016)
23. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S.: Gans Trained by a Two Time-scale Update Rule Converge to a Local Nash Equilibrium. arXiv preprint arXiv:1706.08500 (2017)
24. Liu, J. Y., Chen, Y. H., Yeh, Y. C., Yang, Y. H.: Unconditional Audio Generation with Generative Adversarial Networks and Cycle Regularization. arXiv preprint arXiv:2005.08526 (2020)
25. Kahembwe, E., Ramamoorthy, S.: Lower Dimensional Kernels for Video Discriminators. Neural Networks, vol. 132, pp. 506-520 (2020)
26. Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J.: Long Text Generation via Adversarial Training with Leaked Information. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, No. 1 (2018)
27. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In: Proceedings of the IEEE international conference on computer vision pp. 5907-5915 (2017)

28. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X. and Metaxas, D.N.: Stackgan++: Realistic Image Synthesis with Stacked Generative Adversarial Networks. IEEE transactions on pattern analysis and machine intelligence, vol. 41(8), pp. 1947-1962 (2018)
29. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X. and He, X.: Attngan: Fine-grained Text to Image Generation with Attentional Generative Adversarial Networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1316-1324 (2018)
30. Zhang, Z., Xie, Y., Yang, L.: Photographic Text-to-Image Synthesis with a Hierarchically-nested Adversarial Network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6199-6208 (2018)
31. Karnewar, A. blog: https://medium.com/@animeshsk3/t2f-text-to-face-generation-using-deep-learning-b3b6ba5a5a93 Last accessed 26 June 2020
32. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive Growing of Gans for Improved Quality, Stability, and Variation. arXiv preprint arXiv:1710.10196 (2017)
33. Gatt, A., Tanti, M., Muscat, A., Paggio, P., Farrugia, R.A., Borg, C., Camilleri, K.P., Rosner, M. and Van der Plas, L.: Face2Text: Collecting an Annotated Image Description Corpus for the Generation of Rich Face Descriptions. arXiv preprint arXiv:1803.03827 (2018)
34. Devlin, J., Chang, M. W., Lee, K., Toutanova, K.: Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805 (2018)
35. Cer, D., Yang, Y., Kong, S.Y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., Sung, Y.H.: Universal Sentence Encoder. arXiv preprint arXiv:1803.11175 (2018)
36. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. arXiv preprint arXiv:1705.02364 (2017)
37. Pennington, J., Socher, R., Manning, C. D.: Glove: Global Vectors for Word Representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543 (2014)
38. Nair, V., & Hinton, G. E.: Rectified Linear Units Improve Restricted Boltzmann Machines. In: Icml (2010)
39. Maas, A. L., Hannun, A. Y., Ng, A. Y.: Rectifier Nonlinearities Improve Neural Network Acoustic Models. In: Proc. icml vol. 30, No. 1, p. 3 (2013)
40. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral Normalization for Generative Adversarial Networks. arXiv preprint arXiv:1802.05957 (2018)
41. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot Text-to-Image Generation. arXiv preprint arXiv:2102.12092 (2021)
42. Kingma, D.P. and Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)