Face Generation from Binary Facial Features

Anjandeep Singh Sahni
anjandes@andrew.cmu.edu
Carnegie Mellon University

Dhananjai Sharma
dhananjs@andrew.cmu.edu
Carnegie Mellon University

Kyungzun Rim
krim@andrew.cmu.edu
Carnegie Mellon University

Swapnil Asawa
swap@andrew.cmu.edu
University of Pittsburgh

Abstract

In this project, we focus on generating faces corresponding to certain facial features. We use Generative Adversarial Network (GAN) to pursue our task. Our GAN is an extension of the DCGAN with addition of deep residual learning to learn effectively. We intended to develop a GAN to assist police sketch artists, or to automate the sketch generation task. Here the descriptive facial features are the inputs to the GAN which generates a face corresponding to those features. We also try to control the individual facial features to tweak the changes in the face generated.

1 Introduction

Traditional GAN-based face generators are a lot less informed. They may use loose conditioning on various factors, but it's not as guided as the sketch artist. It creates distribution of faces in general but there is no control on the desired features we would like to have in the faces genrated. In this project, we propose to generate faces directly from binary descriptive facial features, for e.g. the color of eye, similar to what is described to police sketch artist to enable her to create a face. We use the CelebFaces Attributes [1] dataset which has a set of 10K celebrities, approximately 20 images per celebrity. It has 5 facial landmark locations, 40 binary attributes (domains) annotations per image. For this project we assume no missing feature values.

We tried to generate images in such a way that tweaking a parameter shall only tweak that attribute of the image. To do that we are inspired to use the modification of a traditional GAN, which is described in detail in the following sections. The novelty of our project lies in the fact that we do not use any image as the input to the generator network.

1.1 Motivation

This project's success will make it a very useful tool for people to generate face images with ease. One interesting use-case is that such a model can be deployed to generate faces for detecting suspects by the police either by assisting the sketch artists, or by simply automating the task. Another example use-case is generation of attribute based scenery such that it can help reduce the countless hours spent on generating scenery/graphics for games and animated movies. Similarly, different products can be designed just by describing features for example generating purses by describing various aspects of it.

1.2 Objectives

With an aim to develop faces from binary facial features from a GAN, the primary objectives of the project are as follows.

- To develop control over individual facial attributes.
- To generate real-looking faces based on input facial features.

2 Related Work

2.1 DCGAN

A Deep Convolutional Generative Adversarial Network (DCGAN) [2] explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.

- The generator comprises of convolutional-transpose layers, batch norm layers, and ReLU activations. The input to the generator is a latent vector, z, that is drawn from a standard normal distribution and the output is a $3 \times 64 \times 64$ RGB image.
- The discriminator is made up of strided convolution layers, batch norm layers, and LeakyReLU activations. The input to the discriminator is a $3 \times 64 \times 64$ input image and the output is a scalar probability that the input is from the real data distribution.

DCGAN does not establish any relation between the facial features and the generated images. We studied this architecture to use as a baseline.

2.2 StarGAN

Given training data from two different domains, StarGAN [3] model learns to translate images from one domain to the other. StarGAN provides impressive results in terms of image quality as well as control over the facial feature vector.

- Generator takes in as input both the image and target domain label and generates an fake image. The target domain label is spatially replicated and concatenated with the input image.
- Discriminator learns to distinguish between real and fake images and classify the real images to its corresponding domain
- Generator tries to reconstruct the original image from the fake image given the original domain label.
- Generator tries to generate images indistinguishable from real images and classifiable as target domain by Discriminator.
- Aspects of StarGAN that inspired us are the *reconstruction loss* and *domain classification loss*. We use something similar but an easier formulation of these in our model.

However, StarGAN takes a ground truth image as input along with the target domain (to which it will map the ground truth image). This is not suitable for the use-cases we are targeting where the fake image is generated using just the target domain.

2.3 TL-GAN

Transparent Latent-space GAN [4] has also shown significant control over the facial feature vector and is able to generate controlled images. Their model architecture is split into three parts:

- Generator: They use a pre-trained PGGAN generator to create high resolution fake facial images using a random input vector z.
- Feature Extractor: They use a pre-trained domain classifier, which is trained to detect facial features in input images classify them to a 40×1 domain vector.
- Linear Regressor: Using the Generator and Feature Extractor, they are able to get paired data with 1-to-1 mapping between input vector z and the domain of the generated image. They then use a Generalized Linear Model to perform regression between the latent vectors and the features.

As seen above, TL-GAN uses separate pre-trained models. We wanted to explore an end-to-end trainable approach for the same purpose.

2.4 PGGAN

PGGAN [5] focuses on improving the image quality progressively.

- The approach here is to grow both the generator and discriminator progressively: starting from a low resolution, adding new layers that model increasingly fine details as training progresses.
- This both speeds the training up and greatly stabilizes it, giving high quality images.

Like DCGAN, PGGAN does not establish any relation between the facial features and the generated faces. We studied this work as a potential next step to improve the resolution of images generated by our model.

3 Dataset

We used the CelebFaces Attributes (CelebA) [1] dataset for our project. It is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations. Table 1 gives the properties of the dataset. Table 2 shows the 40 binary features of the dataset.

Table 1: CelebA dataset

Number of identities	10177
Number of face images	202599
Number of features	40

Table 2: CelebA features

5 o'clock shadow	No beard
Arched eyebrows	Oval face
Attractive	Pale skin
Bags under eyes	Pointy nose
Bald	Receding hairline
Bangs	Rosy cheeks
Big lips	Sideburns
Big nose	Smiling
Black hair	Straight hair
Blond hair	Wavy hair
Blurry	Wearing earrings
Brown hair	Wearing hat
Bushy eyebrows	Wearing lipstick
Chubby	Wearing necklace
Double chin	Wearing necktie
Eyeglasses	Young
Goatee	Gray hair
Heavy makeup	High cheekbones
Male	Mouth slightly open
Moustache	Narrow eyes

4 Model and Loss Functions

In this section, we describe the GAN that we used to generate faces from binary facial features.

4.1 Baseline Model

We used DCGAN as our baseline as it uses less number of parameters and is simple to comprehend. We expanded DCGAN using deep residual blocks at the lower layers of the generator to extract the input features thoroughly.

4.2 Generator Architecture

Fig. 1 shows the generator G of our model. The input of G is a 40×1 feature vector consisting binary facial features, x. The lower layers are composed of six deep residual blocks which extract the most amount of information from the input. This is followed by transposed-convolutional layers which gradually enlarge the input vector to produce a final 64×64 image with 3 channels, $G(x)$. Our residual blocks use instance normalization. Following the residual blocks, there are five transposed-convolutional layers which use batch normalization and rectified linear unit (ReLU) activation, except for the last layer, which uses only hyperbolic tangent activation.

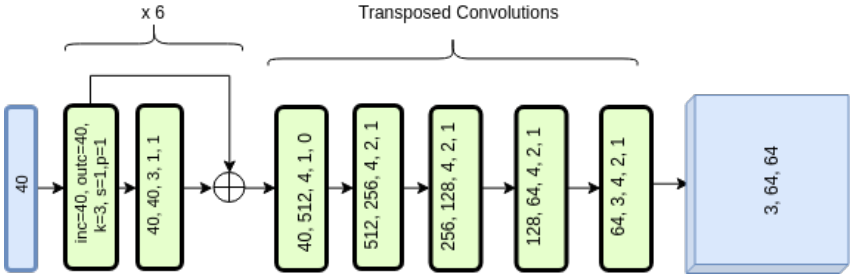

Figure 1: Generator architecture

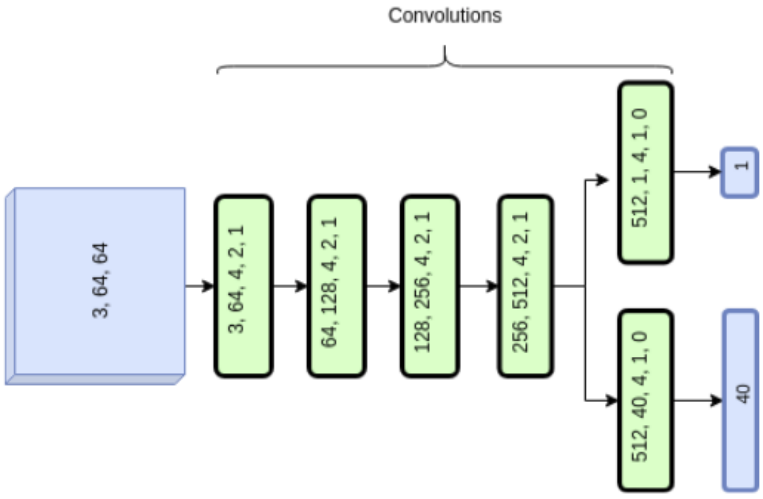

Figure 2: Discriminator architecture

4.3 Discriminator Architecture

Fig. 2 shows the discriminator D of our model. It takes as input a $3 \times 64 \times 64$ image and processes it through four convolutional layers that use batch normalization and leaky ReLU activation, except for the first layer that does not use batch normalization. The resulting tensor will be called as y. D gives two outputs.

- First, it gives a probability that whether the image is real or fake which is achieved by passing y through a convolutional layer followed by sigmoid activation, giving us $D(X)$ and $D(G(x))$, where X is the ground truth image corresponding to the input domain vector x.
- Second, it gives a predicted domain vector of size 40×1, called x', to signify the features present in the input image. The predicted domain x' should ideally be the same as the input domain x.

4.4 Loss Functions

In order to optimize our GAN, we used the following loss functions:

- Adversarial Loss: The generator generates an image $G(x)$ where x is either the input domain or the target domain (as described in the previous section). We use this loss function in the Discriminator D of the network which tries to distinguish between real and fake images. The loss function is as follows:

$$L_{adv} = E[\log D(X)] + E[\log(1 - D(G(x))] \tag{1}$$

  We refer to the term $D(.)$ as a probability distribution of whether the image is real of fake given by D. The generator G tries to minimize this objective, whereas discriminator D tries to maximize it.

- Domain Classification Loss: For an input or a target domain label x, our goal is to first generate an output image $G(x)$ and get the predicted domain label x' from the Discriminator of the network. Thereafter, a natural goal is to minimize the difference between x and x'. We use the following loss function:

$$L_{cls\_net} = L_{cls}(D(x'|X)) + L_{cls}(D(x'|G(x))) \tag{2}$$

  Where $L_{cls}(.)$ refers to binary cross entropy loss between input domain, x, and predicted domain, x'.

- Image Loss: This is $L1$ loss between the fake image, $G(x)$ and the real image X.

$$L_{image} = |real - fake| \tag{3}$$

  We have observed this simple loss function to contribute significantly towards the improvement of image quality as well as the control of generator over the input feature vector.

Table 3 summarizes the loss functions we used. The third column in the table specifies which portion of the network is affected by the corresponding loss function in the first column.

Table 3: Loss functions

Name	Property	G/D
Adversarial Loss	For $fake/real$ images	D, G
Domain Classification Loss	Binary cross entropy between input and predicted domain	D, G
Image Loss	$L1$ norm: $\|real - fake\|$	G

5 Results

The goal of this project was to generate faces which represent the facial features fed into the GAN. This could be achieved by demonstrating control over individual features, in that if a feature is toggled, then only that specific attribute should be changed in the resulting image, keeping all other facets intact. For example, if a feature vector has value 1 for the bangs feature, toggling the same should lead to the image of the same person without bangs.

Fig. 3 shows some of the results that we obtained from our model. It can be seen that toggling a feature indeed changes that feature while keeping the others intact. The first row in each figure shows

the generated image $G(x_1)$ for a given feature vector x_1. The second row shows the generated image $G(x_2)$ for feature vector x_2 where x_2 differs from x_1 in only one feature bit.

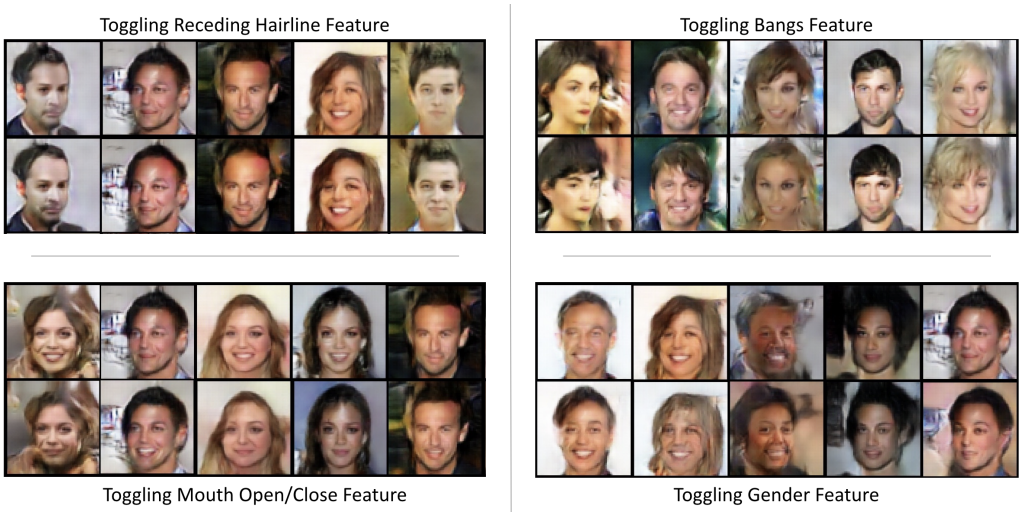

Figure 3: Results of toggling certain facial features

Our model has less than 2 million parameters which makes training much faster as compared to compared to more than 7-8 million parameters in some of the related work. We could generate the above results with no more than four hours of GPU training time.

We were able to achieve the objective of controlling facial features in the generated image for the below mentioned features:

- Bangs
- Gender
- Eye-glasses
- Smiling
- Open mouth
- Chubby face
- Receding hairline

The above seven features showed the most consistent results. In the next section we discuss results for some of the other features which were not controlled as well. We leave that improvement for future work.

6 Conclusions and Further Scope

Our work is able to generate realistic looking faces corresponding to a certain feature vector. As mentioned in the previous section, we could control seven features with high confidence. One detail we want to express is that the dataset contains only 40 features which is not enough to create a unique face, since some of the featues are not very usefule, for example "attractiveness". Below are some of the action items we list for future work to improve upon the architecture of the project:

- Reduce correlation between related features: Some of the facial features in the dataset are highly correlated. For example, the beard feature is highly correlated with the gender (male) feature. Since the dataset does not have any images of females with beard, the generator is currently not able to distinguish between these two features. Upon toggling the beard feature in a female, the model inserts a beard but also tends to make the face more manly.

- Generate multiple faces with same features: Our model generates a single face as yet, whereas in the real world, a police sketch artist, say, might require it to generate a multitude of faces befitting the same description. We are yet to equip our model with that facility. Our hypothesis is that this can be done by adding random noise to the input (for example, concat a 40×1 noise vector to the input vector) to the generator to create a distribution over the most likely faces.
- Explore better architecture: We would also like to explore the possibility of using gender specific GANs. We expect that it can help a great deal as it can ameliorate the situations arising from the cross-correlation among various features. We also plan to make use of the eye landmarks in order to achieve more complex tasks such as changing the eye color.
- Longitudinal control over the features: Currently our model only supports binary control over the facial features. Having a sliding control would help us generate faces with a particular amount of feature, for example, degree of openning/closing the mouth.
- High Resolution Image: Using PGGAN-like generator for better image resolution.

7 Acknowledgements

We would like to thank Professor Bhiksha Ramakrishnan and our mentor Hira Dhamyal for their guidance and feedback throughout the project. We would also like to thank the entire course staff of "11785-Introduction to Deep Learning" at CMU for managing the course.

8 References

[1] Xiaogang Wang Xiaoou Tang Ziwei Liu, Ping Luo. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[2] Soumith Chintala Alec Radford, Luke Metz. Unsupervised representation learning with deep convolutional generative adversarial networks. *CVPR*, 2015.

[3] Munyoung Kim Jung-Woo Ha Sunghun Kim Jaegul Choo Yunjey Choi, Minje Choi. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. September 2018.

[4] Shaobo Guan. Generating custom photo-realistic faces using ai. `https://blog.insightdatascience.com/generating-custom-photo-realistic-faces-using-ai-d170b1b59255`, October 2018. Accessed on 2019-02-15.

[5] Samuli Laine Jaakko Lehtinen Tero Karras, Timo Aila. Progressive growing of gans for improved quality, stability, and variation. October 2017.