Editing Facial Features using GANs

Anmol Jadvani
Department of Computer Science
University of Maryland, College Park
ajadvani@umd.edu

Puneeth Bikkumanla
Department of Computer Science
University of Maryland, College Park
puneethb@umd.edu

Abstract

Generative Adversarial Networks (GANs) have been used widely to synthesize image data. Recently there have been studies on using GANs for style transfer in art and other image to image translations. We aim to build a GAN which uses image data of human faces to generate new images of the same faces with certain facial attributes being edited (e.g. hair style, hair color, etc.) visible in the image. We aim to utilize common existing image to image translation techniques and a popular GAN architecture that has been proven to be successful in image to image translation, CycleGAN. Additionally, if time permits we will also attempt to control the intensity of the facial feature we are trying to generate in an image.

1 Introduction

In this paper, we explore how to synthesize images of real people with different expressions or facial features using a GAN. Facial expression generation has been explored several times in the past and has many applications related to animation, simulation, and human-computer interaction. Current methods include using Computer Vision, GANs, and more. There have been attempts to solve this problem in the past, most involving GANs. We aim to solve the same problem by using a currently existing GAN architecture, CycleGAN [1].

In the past, we have seen CycleGANs having many different applications, including style transfer (applying the texture of one image to another), colorizing photos, and other many more unpaired image translations. We propose using the same architecture towards human faces in order to regenerate an input face with new facial features attached to it. A standard CycleGAN has been proven to be able to translate an image from one specific feature domain to another. In past applications of CycleGAN, we have seen examples such as *Horse2Zebra, Apples2Oranges,* and more.

We found the problem of image to image translation very interesting, and wanted to work on a problem that was more relatable. Many individuals like to imagine themselves with different facial features, whether it be facial hair, hair color, hair style, accessories, etc. We wanted to create a system that allows for individuals to see themselves with specific facial attribute changes, without actually affecting other facial attributes. Although many solutions exist for this problem, we wanted to explore a common image to image translation network, CycleGAN, and incorporate that to solve our problem in mind.

2 Related Work

There has been many work regarding image to image translation, the most notable being CycleGAN [1]. Mentioned in the introduction, CycleGAN is one of the most common implementations of a GAN that uses a source image in a certain feature space to generate the same image, but that belongs in a different feature space. There have been many solutions to the facial attribute editing problem using the same idea of image to image translation that is inspired from CycleGAN. [1] suggests that

Preprint. Work in progress.

incorporating 2 generator networks and 2 discriminator networks allows for this image to image translation. Where one generator and one discriminator pair handles the translation for one feature space to another, while the other pair of generator and discriminator handles the reverse translation. During training in [1], the network takes in a feed image and continuously tries to regenerate the same image by translate the source image to another feature space using one generator and then using the other generator to translate it back to the same source image.

In [2], we see the network architecture AttGAN. [2] has the same problem definition that we are aiming to solve and even has additional implementations, such as intensity control. AttGAN not only generates new images in new feature spaces; however, it has some measure of intensity θ, where a higher θ value corresponds to an image corresponding more to a certain feature space than an image with a lower θ. Similar to the CycleGAN architecture, we see AttGAN having the same idea of using multiple generators, where one generates the source image in a new feature space while the other attempts to regenerate the same source image. Both of these generators end up sharing the loss values with each other in order to ensure the outputs of each are entering the respective target feature space.

In [4], we explore the use of GANs for facial expression generation with expression intensity. [] aims to regenerate faces with different expressions, which is similar to our problem rather it differs by restricting the output only to new facial expression, not facial attributes in general. The architecture used in [] follows very closely to [attgan]; however it introduces an autoencoder to differentiate between the two generators being used. Additionally, there is an additional controller involved in this network F_{ctrl}, which represents the intensity monitor for the expressions. The generator utilizes F_{ctrl} in order to generate images with the proper intensity. This method for intensity control could be useful in our proposed solution, but could be altered in order to represent intensity of any facial feature rather than just facial expressions.

3 Data

3.1 CelebA [3]

This dataset contains 202,599 face images of celebrities and a feature vector of length 36, that contains descriptions for 36 facial features (e.g. Bald, Mustache, Eyeglasses, etc.). Associated with each image is a mapping of “object landmarks," which in this case would be facial structure objects such as nose, eyes, etc. as well as a “label" vector, which loosely follows our definition in the introduction. The label vector for each image is a sequence of -1 or 1 where $label[i] = 1$ represents the image having the attribute corresponding to $feature[i]$, and similarly, $label[i] = -1$ represents the image not having the attribute $feature[i]$. This dataset was used for both training and testing, where the proportion used for training was about twice the size of that used for testing.

4 Method

4.1 Preprocessing

The first step in solving the problem was normalizing the data. First, we wanted the labels to either 0 or 1 because intuitively the labels represent “yes" or “no" for a certain facial feature. This was a very simple process by redefining each $label[i] = \lfloor \frac{label[i]+1}{2} \rfloor$. The images were normalized by first resizing to $(64, 64, 3)$ from their original shape, and each pixel was translated from range $[0, 255]$ to $[0, 1]$.

4.2 Overview

We aim to use the CycleGAN architecture to translate an image of a human face that belongs to one domain: **not** having certain facial feature, to another domain: having that same facial feature. By our problem definition, we want to be able to control many different facial features. A standard CycleGAN will not allow for this.

Let F_x represent feature space x. In the original CycleGAN (depicted in Figure 1), we see to generator networks G_{AB}, which generates an image in F_B using an image in F_A and G_{BA}, which

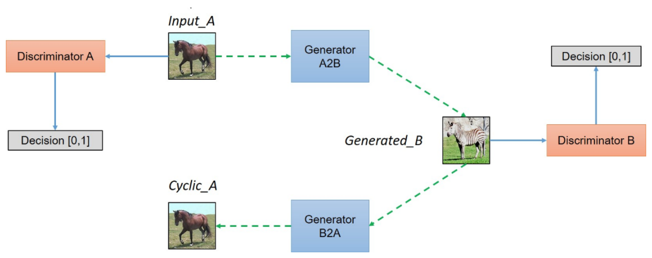

Figure 1: CycleGAN Architecture

generates images in F_A using an image in F_B. Additionally, there exists 2 discriminator networks D_A, where D_A predicts whether any image belongs to F_B and D_B does the same for F_B. To edit this network for our problem, we will have to make several definitions.

First, we will define a vector $features$, which holds all the facial features we are trying to generate. We will assume that we have a feature vector $labels$, where for any image x, $labels_x[i]$ holds a value of 1 if x has the facial feature corresponding to $features[i]$ and 0 otherwise. In our solution, we will be using 2 generator networks for each label indexed i, $G_{TF}[i]$ and $G_{FT}[i]$. $G_{TF}[i]$ will take an image A with $labels_A[i] = 1$ and generate an image A' such that $labels_{A'}[i] = 0$, while $G_{FT}[i]$ will do the same thing but translate from $labels_A[i] = 0$ to $labels_{A'}[i] = 1$.

Similarly each feature space, F_A with index i, had 2 discriminator networks, $D_T[i]$ and $D_F[i]$. $D_T[i]$ would take an image A as an input and output **true** if A corresponded to feature space F_A, while $D_F[i]$ would output **false** if A does not belong to F_A.

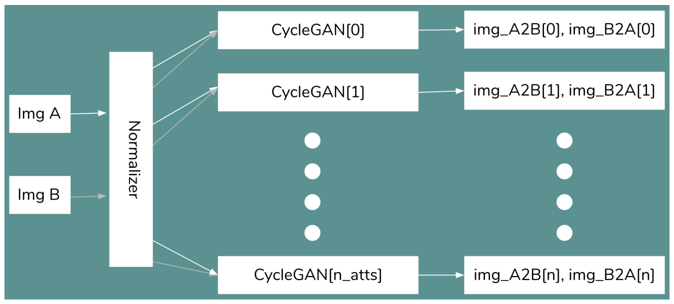

Figure 2: Overall Network

The overall network used is depicted in Figure 2 where n_atts represents how many features we want to train and test the network on. The **CycleGAN** specified in Figure 2 matches the architecture in Figure 1; however the generators and discriminators follow the definition in section 4.3. During training, for each feature space F_B indexed at i, we take two unpaired images A and B, where A does not belong to F_B but B does. The network will train as specified in the original CycleGAN []. During testing, the network will only take in 1 image A, and for each $label_A[i]$ in that image we will generate a new image A' the image with $label_{A'}[i] = 0$ if $label_A[i] = 1$ or generate image A' where $label_{A'}[i] = 1$ if $label_A[i] = 0$.

4.3 Generators & Discriminators

Discriminator	
Conv2D(32,4,2)	ReLU
Conv2D(64,4,2)	ReLU
BatchNormalization	
Conv2D(128,4,2)	ReLU
BatchNormalization	
Conv2D(1,4,1)	

Deconv2D(in, next, filters)
UpSampling2D(2, in)
BatchNormalization
Dropout(0.1)
Concatenate(next)
UpSampling2D(Deconv_3)

ID	Generator	
Conv_1	Conv2D(16,4,2)	ReLU
	BatchNormalization	
Conv_2	Conv2D(32,4,2)	ReLU
	BatchNormalization	
Conv_3	Conv2D(64,4,2)	ReLU
	BatchNormalization	
Conv_4	Conv2D(128,4,2)	ReLU
	BatchNormalization	
Deconv_1	Deconv2D(Conv_4, Conv_3, 128)	
Deconv_2	Deconv2D(Deconv_1, Conv_2, 64)	
Deconv_3	Deconv2D(Deconv_2, Conv_3, 32)	
Deconv_4	UpSampling2D(Deconv_3)	
Conv2D(3,4,1)(Deconv_4)		

Table 1: Generator and Discriminator Definitions

In Table 1, we show the definitions of the generators and discriminators. We have provided a Keras implementation of the model defined, and the following networks mentioned in Table 1 are predefined in the library: *Conv2D, UpSampling2D, BatchNormalization, Dropout, Concatenate*.

4.4 Objective Functions

For simplicity, the generators and discriminator both used the same loss function, Mean Squared Error (MSE), which is defined as follows:

$$\mathcal{L} = \frac{1}{N}\sum_{i=i}^{N}(y^{(i)} - \hat{y}^{(i)})^2$$

MSE is one of the most common loss functions used in deep learning networks, and although it might not have the best application in this case, we used this for our model in the beginning stages to ensure that our model trains properly. Due to time constraints, a proper, more applicable loss function was not derived. Deriving a proper loss function or training the network with a different one is part of our future work.

5 Experiment, Results, & Conclusion

During experimentation, for better computation time, we only tested on 4 of the attributes listed in the CelebA dataset: *Bald, Mustache, Bangs, Eyeglasses*.

5.1 Training

The training sample was a random sample of 1000 images from the CelebA dataset. When training the overall network, we trained each individual CycleGAN with their respective label one at a time. We first trained the *Bald* CycleGAN, followed by *Mustache, Eyeglasses, Bangs*. For each of the label i specified, the network would take 2 images at a time, img_A and img_B where $label_A[i] = 1$ and $label_B[i] = 0$. Since $label_A[i] \neq label_B[i]$, we know that these 2 images must be unpaired, and therefore we trained our generator to take both add or remove $label[i]$, depending on which of the two generators is being used. The entire model was only trained for 1 epoch due to constraints, specifically time constraints (30 hours for 1 epoch) and computation constraints (used all of Google Cloud credits).

5.1.1 Training Results

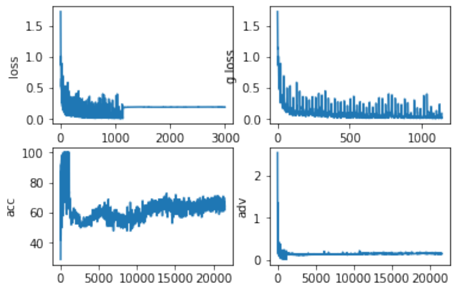

Figure 3: Label = Bald

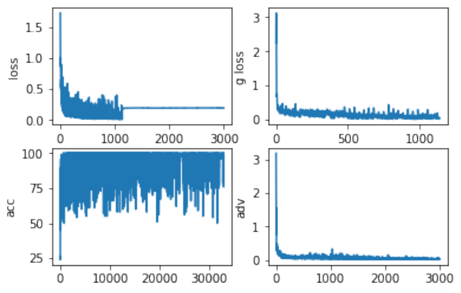

Figure 4: Label = Mustache

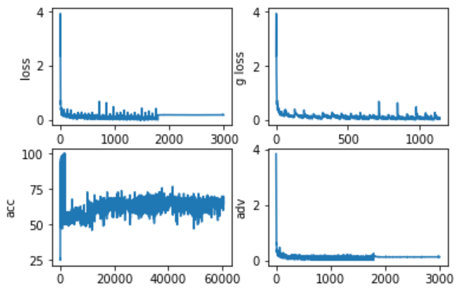

Figure 5: Label = Eyeglasses

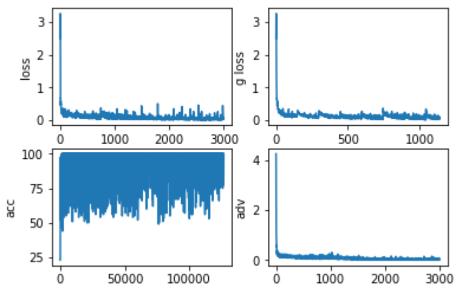

Figure 6: Label = Bangs

In Figures 3 through 6, we see the loss curves for the discriminator and the generator networks as well as the accuracy curve for the discriminator for each of the 4 labels we trained on. In Figure 4, we see really good loss curves for both the discriminators and the generators. Although the discriminator for the *Mustache* label seems to have many drops in the accuracy for the discriminator, these are actually due to outliers. Each drop in the accuracy curve was only because of one faulty iteration, and the ending accuracy for the discriminator network was nearly 100%. The loss curves also had a relatively good result for this specific label, where the adversarial loss and the discriminator loss both ended up less than 0.01 by the end of training. We received similar results for the *Bangs* label.

However, for the remaining two labels that were trained on, the generators ended up having a loss of **nan** in the middle of training. These **nan** values are obviously not depicted in the figures above. Because of these loss values, we saw that the generator ended up not working properly in the testing stage.

5.2 Testing

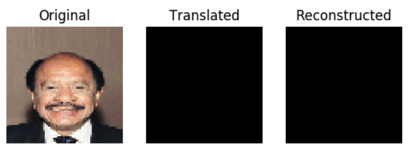

Figure 7: Failed Result with Label = Bald

Due to the training issue mention in the previous section, there were a lot of testing issues that arose for the labels *Bald, Eyeglasses*. Since the loss values of the generators his a **nan** value, the generator ended up only generating blank images. A sample is shown in Figure X.

The more successful generations existed in the labels *Mustache* and *Bangs*. For both of these results, although the results were not perfect, we see that the generator essentially recolored the mustache area or bangs area to closer match the skin tone of the subject.

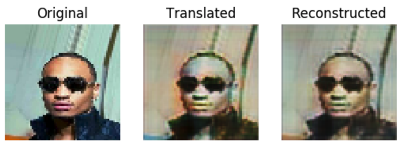

Figure 8: Result with Label = Mustache

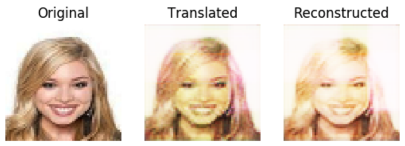

Figure 9: Result with Label = Bangs

We see these results in Figures X and Y. Likely, with a longer training cycle as well as a training cycle that includes more samples in the dataset, the generator would perform a lot better than in the samples provided.

6 Future Work

There is a lot of work to be done for this model. As seen in the testing results, the model does not perform as well as we had hoped for on some of the labels. Our short term future goals for this project mainly revolve around ensuring that the model works for more of the attributes specified in the CelebA dataset. To start this off, we would have to tune the GAN such that the loss values do not hit a **nan**. The most likely cause for the **nan** values would be the loss functions used for the generator networks. So, we would like to come up with better loss functions than those are currently used and then follow with retraining the network. In addition to this, when retraining the model with a different loss function, it might be more beneficial to train the discriminator networks separately rather than alongside the generators.

Another short term goal of ours would be to edit the CycleGAN architecture rather than copying it exactly. In the standard CycleGAN, the network is used to transfer an image or vector from feature space A to feature space B, which requires 2 discriminators, D_A and D_B. Since our solution to the problem only aims to transfer an image from feature space A to feature space **not** A, only one discriminator would be necessary. This was realized well after our implementation was complete and it was unreasonable to change the architecture and retrain the model due to time constraints.

After having a successful network, there are also some longer term goals we have for this project. In past projects, such as in [], we have seen a measure of *intensity* fed into the input. This intensity level describes how severe an attribute will be added or removed from a certain face. Our long term goal for this project would be to add this implementation to our network as well. This would require us to add thresholds to our generator and discriminator networks. Additionally, rather than labels being 0 or 1, we would need the label to take on a range between [0, 1], where 0 represents the attribute not existing, and 1 representing the attribute existing to the highest degree. After this, we will need to introduce another input dimension to each generator network, which represents the intensity of the feature we are generating or removing.

Miscellaneous

1. **Link to Implementation:** https://github.com/ajvani/cmsc498L-final
    - Instruction for downloading dataset, training, and testing is specified in the README.
2. **Link to Presentation:** Google Sheets
    - Link also available in Github README

References

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR, abs/1703.10593, 2017.

[2] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen. Arbitrary facial attribute editing: Only change what you want. arXiv preprint arXiv:1711.10678, 2017.

[3] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. Retrieved August, 15:2018, 2018.

[4] Ding, H., Sricharan, K., Chellappa, R. ExprGAN: Facial Expression Editing with Controllable Expression Intensity.