**Using GANs to Style Human Faces.**

**Abstract**

**1. Introduction**

Breakthroughs in deep learning have led to major advancements in the field of style transfer. Style transfer is the technique of reconstructing images in the style of another image. However, style transfer is not just restricted to images [1]. The use of Adversarial networks has found much success in translating the style of video footage, text-image and more. In this paper we look to explore the use of Generative Adversarial Networks (GANs) in order to perform style transfer on human faces. We seek to discover the cross-domain relationships across different attributes such as gender, age, facial features and expressions. We look at three versions of GANs. The first being a standard GAN, secondly a standard GAN with reconstruction loss and finally we use a Discovery GAN (DiscoGAN) where we allow bijective mapping of our images.

**2. Motivation**

For our group project we wanted to explore an idea which was quite new in the field of deep learning. Whilst we certainly had the option to perform style transfer with Convolutional neural networks, we felt that attempting to do it using unsupervised methods will certainly be more challenging and intriguing. To make things more interesting we chose to work with human faces to see answer questions such as; Can we determine how people will age as they grow older? How can we use these methods to change human expression? and finally can a machine learn to generate features on people who has never had those features before? If it is indeed possible to perform all these tasks using such techniques, then we could produce some really novel applications such as reconstructing some one’s youth, or mapping out how a person will age. We could even use these techniques in the field of cosmetic surgery, giving patients the freedom to pick the facial attributes they desire and generate a before and after image before the procedure.

**3. Generative Adversarial Networks**

GANs are very powerful and malleable models. They are composed of a Generator and a Discriminator. The Generator will constantly attempt to generate images by applying features of its target image data to the input image. For example, for a Generator trained with target images of female, when giving as input an image of a male, it will produce a new image of a man with female features. Whilst the Discriminator will try to determine whether or not the Generator’s images look like the real target image. In our previous example, the Discriminator would determine if the fake « female » image is different from a real female image. Hence, these two networks work in an adversarial relationship against one another with the generator attempting to trick the Discriminator.

Since the inception of GANs, there have been many variations of the model such as the CycleGAN or LSGAN. These variations help to alleviate some of the problems that GANs are susceptible to. Although the network architecture may not change the overall performance drastically, using different loss functions or hyper parameters can lead to much better results in terms of image quality. LSGANs for example use a least squared loss and the images produced by LSGANs are remarkably better than that of a normal GAN or DCGAN, this is due to the fact that the Discriminator conventionally uses a sigmoid cross entropy loss, which leads to the problem of vanishing gradients [2].

**4. Dataset**

The dataset used in this paper was the CelebA dataset [3] which is a dataset curated by the Chinese University of Honk-kong. The dataset consists of over 10K celebrity identities, with over 200K headshot images. The dataset has over 40 different attributes – ‘wearing a hat’, ‘pointy nose’, ‘Male’, ‘Female’, ’Smiles’ etc.

One major issue we had was class balancing. As a celebrity could fit into more than one category, they could be male, whilst wearing a hat and smiling, this meant that when it came to sorting data into two separate categories for style transfer certain features could start to dominate the algorithm. For example, when we wanted to apply smiles to non-smiling faces, we found out smiling pictures consisted of more females than males so when transferring smiles to non-smiles and vice-versa, it led to a result where all the non-smiling images were male and the smiling images were female. In this study, we don’t want to change more than one feature of the image. We overcame this problem by balancing the datasets as much as possible by removing the excess of smiling female images and replacing them with male smiling images and did the same to the non smiling category.

Another problem we faced was the size of the dataset, although having over 200K images will lead to less overfitting in the model unfortunately, it proved very costly to store the dataset. When unzipping the images, a lot of time was lost in loading the dataset before beginning to sort the data into our required categories. Conveniently the dataset had a text file which gave each image a set of attributes, which we could use to sort them into their corresponding category. It was very useful, as we wrote a script to help us quickly sift through the dataset and categorize our data into our required class.

We considered the IMDB-WIKI dataset [4] which was in many ways similar to the CelebA. However, some of the images were corrupted and caused the code to throw errors in unexpected places, although the solution was to remove all of the corrupted images because the dataset was too large, it proved too time costly to clean the data in comparison to the CelebA dataset.

**5. Models**

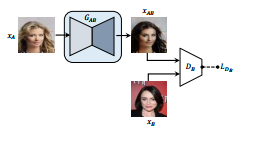
**5.1 Plain GAN**

**5.1a Model.**

Our first basic GAN consisted of one generator and one discriminator as shown by Figure 1. The Generator consisted of 4 convolutional layers and 4 de-convolutional layers with the final layer having a relu activation function. It takes an image as an input and outputs an image. Whilst the Discriminator consisted of 4 convolutional layers with a sigmoid activation function on the final layer. It takes two images as an input and outputs a value between 0 and 1. We used a stride size of 5 with zero padding and batch-normalisation in every layer. We also applied dropout to the first four layers of the convolutional portion of the generator. Furthermore, we scaled all our images down to

64x64 pixels as it’s very difficult for GAN’s to converge on much larger image sizes [5].

Our approach can be visually demonstrated by Figure 1, we took an input image and fed it through the generator and attempted to produce the image , we would then feed both these images through to the discriminator which would calculate how well our generator replicated the target image. This would be the generator loss. We would then attempt to minimise the loss of the generator to get better quality images.

Figure 1.

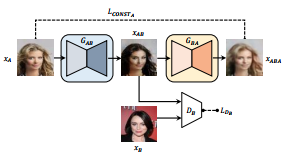
**5.1b Analysis.**

This model outputs acceptable images but we face a problem. During training, the generator learns the mapping from domain A to domain B but there is no constraint to make learn the reconstruction mapping from B to A. With a standard GAN, the mapping is one-directional. In other words, the mapping is an injection, not bijection, and one-to-one correspondence is not guaranteed, which is a requirement for a cross-domain transfer.

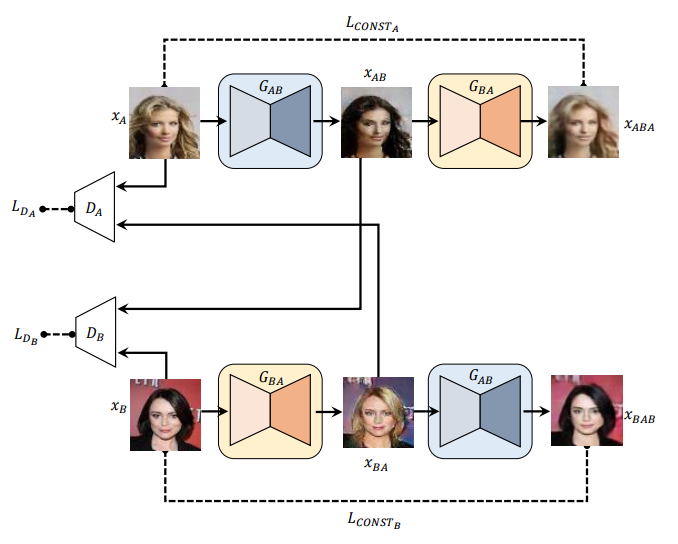
**5.2 GAN with Reconstruction loss 5.2a Model**

To improve the model, we added a second generator as demonstrated by Figure 2. First, we feed in an image into generator then we take the generated image and feed it into a second generator which will attempt to reconstruct the original image . This allows us to constraint a mapping from B to A.

Figure 2.

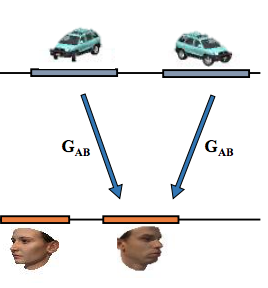
For this model we did not change any of the hyper parameters. The loss function for is now calculated as:

(1)

Where is the original loss of our first generator and is our reconstruction loss obtained from our second generator. We add these two losses to get which is our new loss function that we attempt to minimise.

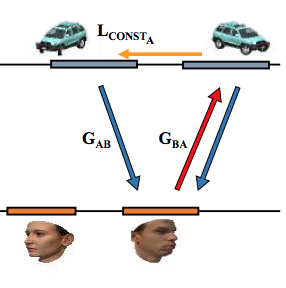
To calculate the reconstruction loss , we take the difference between the image generated by the second generator and the original input image. To do this we need a distance function [6], in our case we used the MSE.

**5.2b Analysis**

Let’s use Figure 3 to illustrate the mode collapse problem.

In a standard GAN, does map cars (A) to faces (B) but does not take into account the orientation, which is a desired property, in our case. We want to make retain more information about the input data. This would allow to reconstruct A from B with enough accuracy. That way, we would make map A to B - cars to faces - and map the orientation, as wanted.

Although adding a reconstruction loss forces the reconstructed sample to match the original, in practice it does not resolve mode-collapsing. As still maps left and right orientations to a single orientation in the face space, will only be able to map B to a single orientation in A (Figure 4).



**5.2 Discovery GAN**

**5.2a Model**

As introduced in this paper [6], DiscoGAN couples the GAN with reconstruction loss model into two layers (Figure 5).

Each of the coupled layers learns the mapping from a domain to an other, and the associated reconstruction mapping. The two layers are trained simultaneously. Of course, and in the two layers are the same generators and share their parameters. The generated images and are each given as input to two separate discriminators and , respectively. In each layer, the reconstruction loss is computed ( and). This allows us to implement the reconstruction for both domains.

The new generator and discriminator losses are:

(7)

=

which is the sum of GAN loss and reconstruction loss for each layer.

(8)

which is the sum of the discriminator losses from and .

**5.2b Analysis**

With this model, we obtain a bijective mapping and a one-to-one correspondence. A cross-domain relation can be fully discovered.

**5.2c Optimization**

After training the model with different hyper-parameters, we finally kept this configuration: […]

**6. Experiment**

**6.1 Plain GAN**

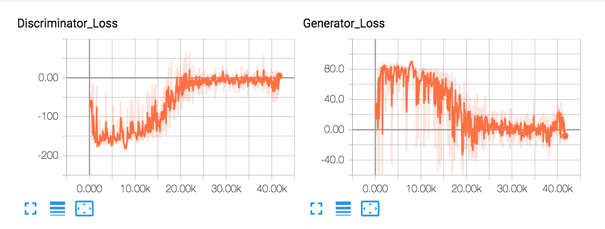
The basic GAN implementation was trained for 30 epochs using two-thousand images divided into ‘Male’ and ‘Female’ feature classes.

The Tensorflow library provides a suite of visualisation tools called Tensorboard, which was adopted within the implementation. Using Tensorboard, the values of the generator and discriminator losses were captured at each step of the training process, and are presented below in <fig#>

As can be seen from <fig#>, the loss function was quickly minimised during the first few epochs of training, and a corresponding improvement in image quality from white noise to face-like representations (perhaps helped a little by the pareidolia phenomenon) is seen. This was followed by no real

A sample batch of generated images from the final epoch is shown below in <fig#>. Although it is clear that facial features were successfully generated, the quality is poor and there is no discernable style transfer; indeed the representation is not sufficiently clear to classify any of the images from inspection.

**6.2 GAN with Reconstruction Loss**

The second implementation, of the GAN with reconstruction loss, was also trained on the same data set for 30 epochs. The results were comparable to those of the vanilla GAN implementation. As can be seen in <fig> the discriminator and generator losses follow much the same trajectory, of rapid early gains followed by oscillatory behaviour one the generator consistently ‘fools’ the discriminator.

A sample batch of generated images from the final epoch is shown in <fig>. As before, the results fail to visually represent features of either the source or target classification set.

The issue appears to be that the generator learns too quickly for the discriminator; (or the discriminator is not discriminatory enough??) Once the training reaches a generator loss of zero, the discriminator is unable to discern real target images from the generated set. This could be to do the with the discriminator loss not being accurate enough? Or in so much variation (64 image batches) then the difference between these features are two tiny.. in fact don’t we have 64x64 images? So each image is 8x8? A final possibility is that it compares the generated image against simply one random image from the target sample. Better would be to compare against an average of some kind, but not sure how to implement this..

Failures; ‘mode collapse’

**6.3 DiscoveryGAN**

**7. Conclusion**

Evaluation, Limitations and Future Work

<https://arxiv.org/pdf/1703.10593.pdf> **[1]**

<https://arxiv.org/abs/1611.04076> **[2]**

**[3]** <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

**[4]**

<https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/>

**[5]**

<https://www.quora.com/Do-generative-adversarial-networks-always-converge>

**[6]**

<https://arxiv.org/pdf/1703.05192.pdf>