Gender Change of People's Faces using CycleGAN

Kishalaya De, Nipun Manral, Ranga Sai Shreyas Manchikanti
kishalad@usc.edu, manral@usc.edu, rmanchik@usc.edu

Contents

List of Figures

List of Tables

1 Abstract

Modern surveillance systems that rely on recognising people based on faces are not very robust to facial disguises. To improve the performance of such systems, we propose an approach to generate images that the facial recognition software can use to train. We generate gender changed images of people's faces. To produce such images, we use the CycleGAN architecture to perform male to female domain transformation and vice-versa. The CycleGAN architecture excels in generating images without paired images in the two domains which is appropriate for our application since we generally do not have gender disguised images of the people. We implemented the model in keras and trained the system on CelebA dataset to obtain realistic looking gender changed faces. Then we experimented on the model by changing losses such as identity loss used in training. The results are thoroughly examined and we found that identity loss is essential for producing authentic images.

2 Introduction

2.1 Problem Statement and Goals

The goal of this project is to change the gender of a person's face using a CycleGAN architecture. For this purpose, we first implement the model in Keras and train the model on the Monet2Photo dataset to validate it. Next, we train the same CycleGAN model on the CelebA dataset with identity loss and train a different model without identity loss to compare the results. By analyzing the different results, we conclude with the best model that generates realistic images and further explore future improvements.

2.2 Application

Modern surveillance system based on facial recognition suffer from a few limitations. They are still not robust to disguises such as - make-up, glasses and gender change as evident from the news article [1]. The proposed CycleGAN architecture can overcome this limitation by generating examples of corner cases - a man in a woman's disguise and vice versa. Training the surveillance systems with such images will help detect and identify criminals in diguises.

2.3 Related Work

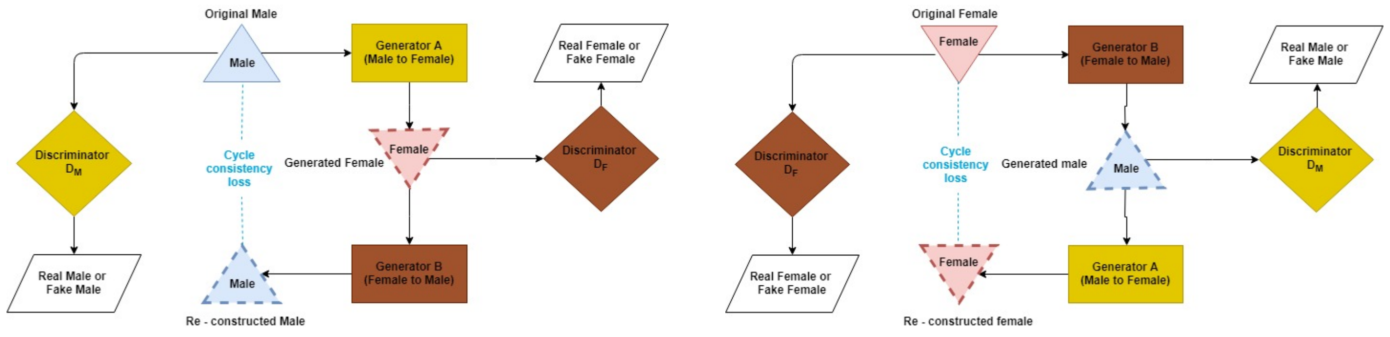
The domain to domain translation is not a novel topic and has already been implemented in the Pix2Pix network which utilizes Conditional GANs [2] to achieve this goal.

Let (x_1, y_1) be an original image pair in the two domains X and Y respectively. The conditional GAN (cGAN) differs from vanilla GAN in a way that the generator of cGAN takes the original image x_1 in addition to the noise. This conditions the generator to produce an image with a distribution similar to that of the original image x_1. The generator produces a fake image in domain Y called $\widehat{y_1}$. In order to train the discriminator, we need to train it on both $\widehat{y_1}$ and y_1.

From this approach, we can observe that a major limitation of cGANs is that it requires paired training images of the same object in both the domains which might not be always available for cross-domain transformation tasks. This limitation can be overcome using Cycle GANs in the absence of paired training images.

3 Architecture

The CycleGAN is a type of GAN that is used for cross-domain transfer tasks. For our project, the CycleGAN is used to transform an image from a domain M (Male images) to a different domain F (Female images), without needing corresponding individual image pairs in both the domains. To obtain this fidelity, the Cycle-GAN consists of a network having two discriminators and two generators.



(a) Male to Female (b) Female to Male

Figure 3.1: Model Architecture

3.1 Generator

The CycleGAN consists consists of two generator networks A and B which try to generate an image from the source domain and convert it into an image that is similar to the target domain. The generator's architecture is similar to the network used in neural style transfer which consists of an encoding block, a transformation block and a decoding block. The encoder extracts the input features using convolution layers with stride 2. The transformation block transforms the features using residual blocks. The decoder decodes the transformed features using convolution layers with fractional stride $\frac{1}{2}$.

Generator A: The generator A learns a mapping from Domain M (Male) to Domain F (Female).

$$A : M \rightarrow F$$

Generator B: The generator B learns a mapping from Domain F (Female) to Domain M(male).

$$B : F \rightarrow M$$

3.1.1 Downsampling blocks

Layer Name	**Hyperparameters**	**Input Shape**	**Output Shape**
2D convolution layer	filters=32, kernel_size=7, strides=1, padding='same'	(128, 128, 3)	(128, 128, 64)
Instance normalization layer	axis=1	(128, 128, 64)	(128, 128, 64)
Activation layer	activation='relu'	(128, 128, 64)	(128, 128, 64)
2D convolution layer	filters=64, kernel_size=3, strides=2, padding='same'	(128, 128, 64)	(64, 64, 128)
Instance normalization layer	axis=1	(64, 64, 128)	(64, 64, 128)
Activation layer	activation='relu'	(64, 64, 128)	(64, 64, 128)
2D convolution layer	filters=128, kernel_size=3, strides=2, padding='same'	(64, 64, 128)	(32, 32, 256)
Instance normalization layer	axis=1	(32, 32, 256)	(32, 32, 256)
Activation layer	activation='relu'	(32, 32, 256)	(32, 32, 256)

Table 1: Downsampling block configuration

3.1.2 Residual - Transformation

Layer Name	Hyperparameters	Input Shape	Output Shape
2D convolution layer	filters=256, kernel_size=3, strides=1, padding=‘same’	(32, 32, 256)	(32, 32, 256)
Batch normalization Layer	axis=3, momentum=0.9 epsilon=1e-5	(32, 32, 256)	(32, 32, 256)
2D convolution layer	filters=256, kernel_size=3, strides=1, padding=‘same’	(32, 32, 256)	(32, 32, 256)
Batch normalization layer	axis=3, momentum=0.9 epsilon=1e-5	(32, 32, 256)	(32, 32, 256)
Addition layer	None	(32, 32, 128)	(32, 32, 128)

Table 2: Transformation block configuration

3.1.3 Upsampling block

Layer Name	Hyperparameters	Input Shape	Output Shape
Transpose 2D convolution layer	filters=128, kernel_size=3,strides=2, padding=‘same’, use_bias=False	(32, 32, 256)	(64, 64, 128)
Instance normalization layer	axis=1	(64, 64, 128)	(64, 64, 128)
Activation Layer	activation=‘relu’	(64, 64, 128)	(64, 64, 128)
Transpose 2D convolution layer	filters=64, kernel_size=3, strides=2, padding=‘same’, use_bias=False	(64, 64, 128)	(128, 128, 64)
Instance normalization layer	axis=1	(128, 128, 64)	(128, 128, 64)
Activation Layer	activation=‘relu’	(128, 128, 64)	(128, 128, 64)
2D convolution layer	filters=3, kernel_size=7, strides=1, padding=‘same’, activation=‘tanh’	(128, 128, 64)	(128, 128, 3)

Table 3: Upsampling block configuration

3.2 Discriminator

The CycleGAN consists of two discriminator networks D_M and D_F which try to discriminate between the generated images(fake images) and real images. The architecture of a discriminator consists of a deep convolution neural network.

Discriminator D_M: The discriminator D_M tries to distinguish between the male images generated by Generator B [B(F)] and the original male images belonging to domain M.

Discriminator D_F: The discriminator D_F tries to distinguish between the female images generated by Generator A [A(M)] and the original female images belonging to domain F.

Layer Name	**Hyperparameters**	**Input Shape**	**Output Shape**
Input layer	none	(128, 128, 3)	(128, 128, 3)
Zero Padding 2D layer	padding(1, 1)	(128, 128, 3)	(130, 130, 3)
2D convolution layer	filters=64, kernel_size=4, strides=2, padding='valid'	(130, 130, 3)	(64, 64, 64)
Activation layer	activation='leakyrelu', alpha=0.2	(64, 64, 64)	(64, 64, 64)
Zero Padding 2D layer	padding(1, 1)	(64, 64, 64)	(66, 66, 64)
2D convolution layer	filters=128, kernel_size=4, strides=2, padding='valid'	(66, 66, 64)	(32, 32, 128)
Instance normalizationlayer	axis=1	(32, 32, 128)	(32, 32, 128)
Activation layer	activation='leakyrelu', alpha=0.2	(32, 32, 128)	(32, 32, 128)
ZeroPadding2Dlayer	padding(1, 1)	(32, 32, 128)	(34, 34, 128)
2D convolutionlayer	filters=256, kernel_size=4, strides=2, padding='valid'	(34, 34, 128)	(16, 16, 256)
Instance normalization layer	axis=1	(16, 16, 256)	(16, 16, 256)
Activation layer	activation='leakyrelu', alpha=0.2	(16, 16, 256)	(16, 16, 256)
ZeroPadding2D layer	padding(1, 1)	(16, 16, 256)	(18, 18, 256)
2D convolutionlayer	filters=512, kernel_size=4, strides=2, padding='valid'	(18, 18, 256)	(8, 8, 512)
Instance normalizationlayer	axis=1	(8, 8, 512)	(8, 8, 512)
Activation layer	activation='leakyrelu', alpha=0.2	(8, 8, 512)	(8, 8, 512)
ZeroPadding2D layer	padding(1, 1)	(8, 8, 512)	(10, 10, 512)
2D convolutionlayer	filters=1, kernel_size=4, strides=1, padding='valid', activation='sigmoid'	(10, 10, 512)	(7, 7, 1)

Table 4: Discriminator configuration

3.3 Training Objective

The original paper defines a weighted sum of two loss functions - Adversarial Loss and Cycle Consistency Loss.

Adversarial Loss

The adversarial loss is the loss between the real image and the generated fake image.
Adversarial Loss for $A : M \to F$

$$L_{GAN}(A, D_F, M, F) = E_{f\sim P_{data}(f)}[logD_F(f)] + E_{m\sim P_{data}(m)}[log(1 - D_F(A(m)))]$$

Cycle Consistency Loss

Since adversarial losses alone cannot guarantee that the generated image has the same distribution as the original image, we need an additional loss called the cycle consistency loss. This loss reduces the space of possible mappings between domain M and F that the generator can produce.

$$L_{cyc}(A, B) = E_{m\sim P_{data}(m)}[\|B(A(m)) - m\|_1] + E_{f\sim P_{data}(f)}[\|A(B(f)) - f\|_1]$$

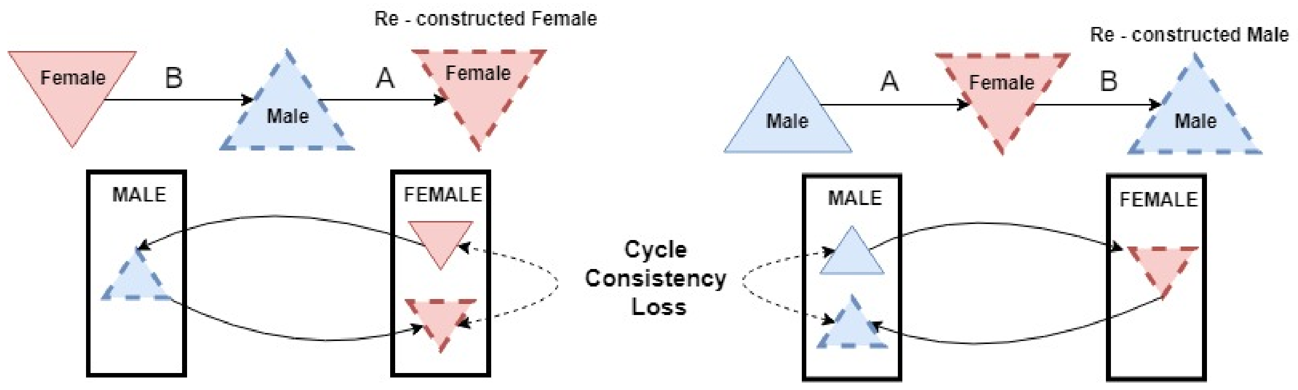


Figure 3.2: Consistency Loss

4 Novelty

- We have implemented the code for the architecture and computing loss from scratch in Keras and tested the validity on Monet2Photo datasets.
- We have developed a novel application by training a model for gender swapping using CycleGAN architecture.

- We have experimented with identity loss and its effect on our generated images which has been used in the photo generation from paintings application [3]. Identity loss should ensure that if the source domain and the target domain are the same, then the generated images should be the same as the input images in order to avoid unnecessary color tinting.

$$L_{identity}(A,B) = E_{m\sim P_{data}(m)}[\|B(m)-m\|_1] + E_{f\sim P_{data}(f)}[\|A(f)-f\|_1] \tag{1}$$

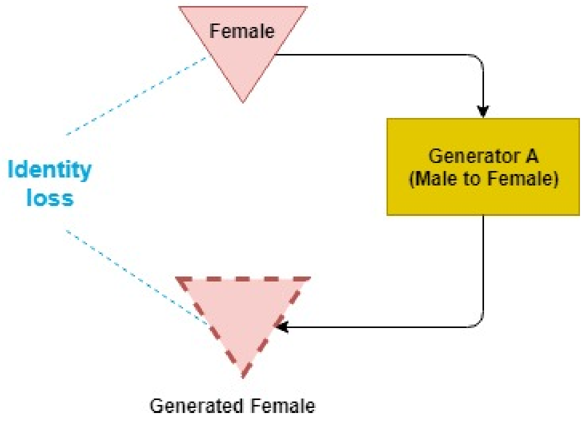


Figure 4.1: Identity Loss

5 Implementation and Training

5.1 Datasets

5.1.1 Monet2Photo

The Monet2Photo dataset consists of 2 sets of images - one set contains images of landscape paintings by the artist Monet and the other set contains photo-realistic landscape images. Each image is of size 256x256 pixels. We used about one thousand images of each set for training our network. This data is open source and is made available by the Berkeley AI research (BAIR) lab at UC Berkeley.

5.1.2 CelebA

The CelebFaces Attributes Dataset(CelebA) [4] has about two hundred thousand images of ten thousand unique celebrity faces. The dataset contains around 40 different face attributes with different poses, background clutter and landmark locations including the gender of the person.

5.2 Pre-processing

The gender attribute of the CelebA dataset was used to categorize the images into male and female images. Next, the CelebA dataset images were re-sized from 256x256 dimensions to 128x128 dimension color image for training. Since there were a lot of variations in the images, we visually inspected most of the images to remove images which had bad lighting or awkward face poses which caused it difficult to observe the facial features.

5.3 Training

The code for implementing the CycleGAN model and training the network was implemented from scratch in Keras. The model was first trained on the Monet2Photo dataset to check the validity of the implementation. Each epoch had about one thousand images in each domain. The training was done for 50 epochs with batch size 1 using an Adam optimiser with a learning rate of 0.0002. The training was performed on an AWS instance p2.xlarge which has a NVIDIA Tesla K80 GPU. The results obtained are analyzed and discussed in the next section.

The following training configurations were explored to train on the CelebA dataset.

- Generator with ResNet-6 block for transformation with weight of cycle consistency loss λ equal to 10 and for identity loss equal to 10
- Generator with ResNet-6 block for transformation with weight of cycle consistency loss λ equal to 10 and for identity loss equal to 5
- Generator with ResNet-9 block for transformation with weight of cycle consistency loss λ equal to 10 and for identity loss equal to 5
- Generator with ResNet-9 block for transformation with weight of cycle consistency loss λ equal to 10 and for identity loss equal to 10
- Generator with ResNet-9 block for transformation with weight of cycle consistency loss λ equal to 5 and for identity loss equal to 5

All the above mentioned training configurations were run for 40 epochs with a batch size of 1 using an Adam optimiser and learning rate 0.0002. The training was executed on an AWS instance p2.xlarge machine and the results of these configurations are examined and discussed in the next section.

6 Results

6.1 Evaluation on Monet2Photo dataset

We tested our keras implementation of CycleGAN on the Monet2Photo dataset in order to evaluate its performance. The model was trained for 50 epochs and the results are as follows:

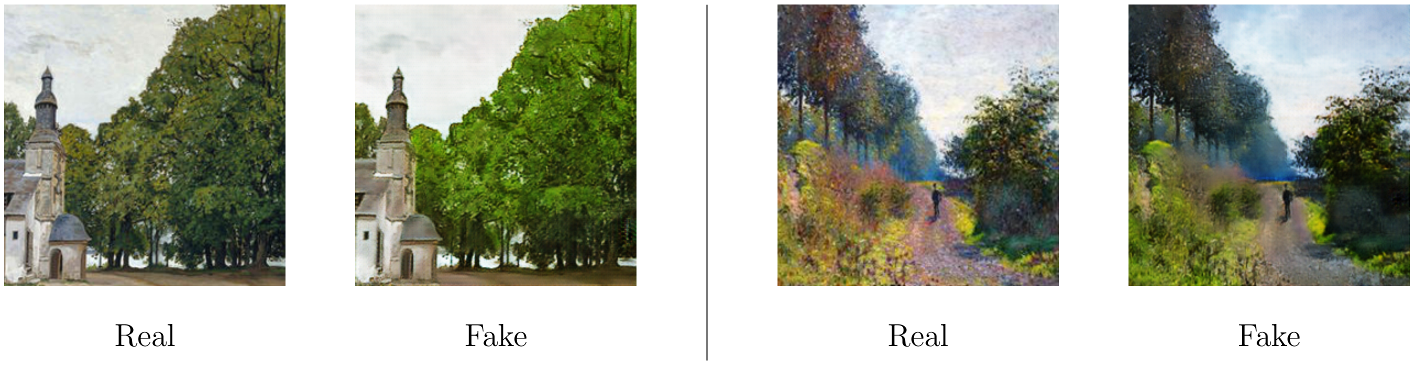


Figure 6.1: Monet → Photo

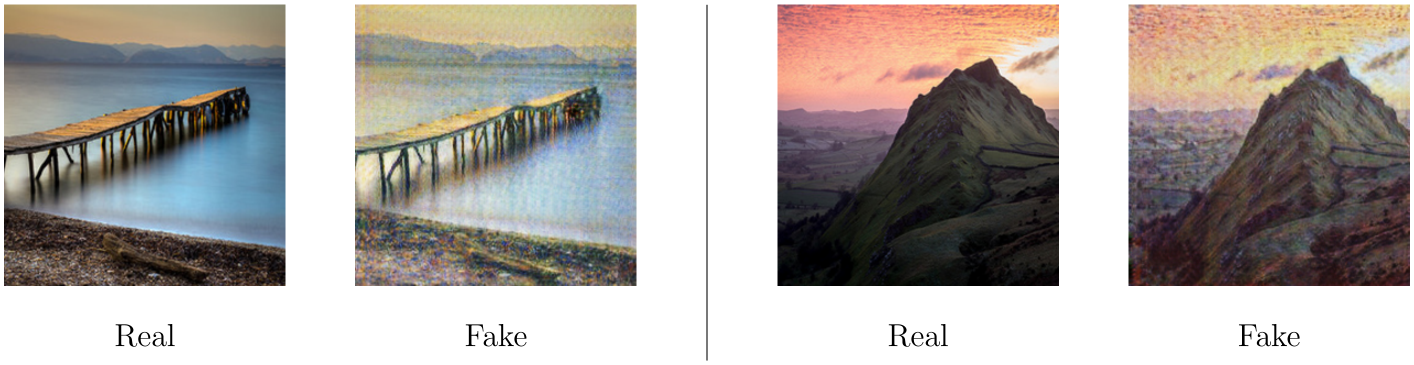


Figure 6.2: Photo → Monet

For the Monet2Photo conversion in Figure 6.1, we can notice that the trees, the sky and the background look very realistic in the generated images. In figure 6.2, the sky in the fake image for the first pair and the bridge/water in the second pair have a very significant change in color tone which is highly similar to Monet's painting style. The results obtained were very reasonable, which implied that our keras implementation of the CycleGAN architecture worked as expected.

6.2 Evaluation on CelebA dataset

After experimenting with the various training configurations discussed in section 5.3, the best results were obtained with the ResNet-9 transformation block in the generator. The λ parameter for cycle consistency loss and the identity loss provided the best results when they were set as 10 and 5 respectively. We ran the code on CelebA dataset with this configuration to obtain the results as shown below:

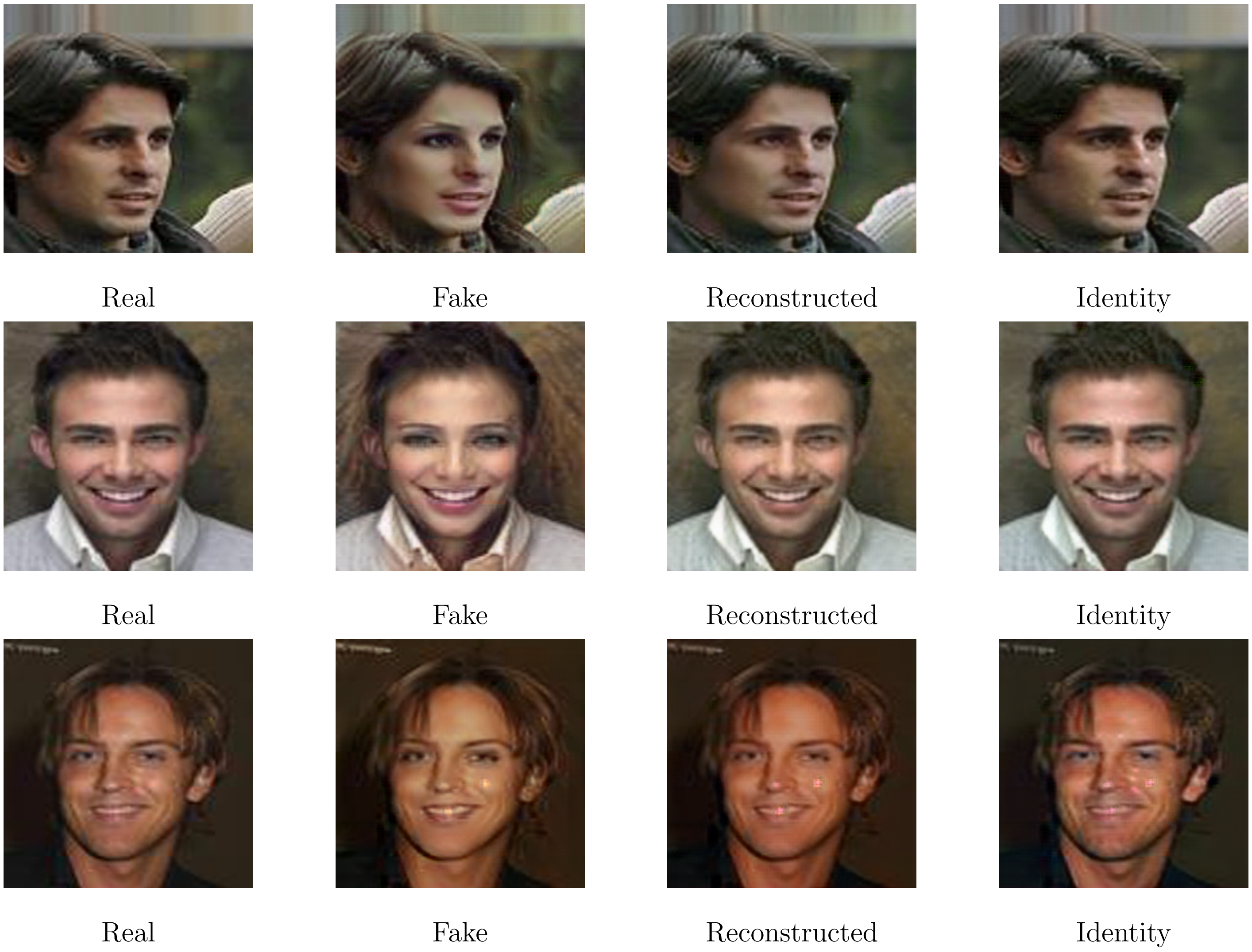


Figure 6.3: Male $\rightarrow$ Female

In the results of the male to female conversion, we notice that the generated fake images look much more feminine in comparison to the original input image. The changes in facial features are noticeable around the lips, nose, cheeks and the color tone. Also, the reconstructed images look almost the same as the input images which shows that the cyclic consistency loss is getting minimized as expected.

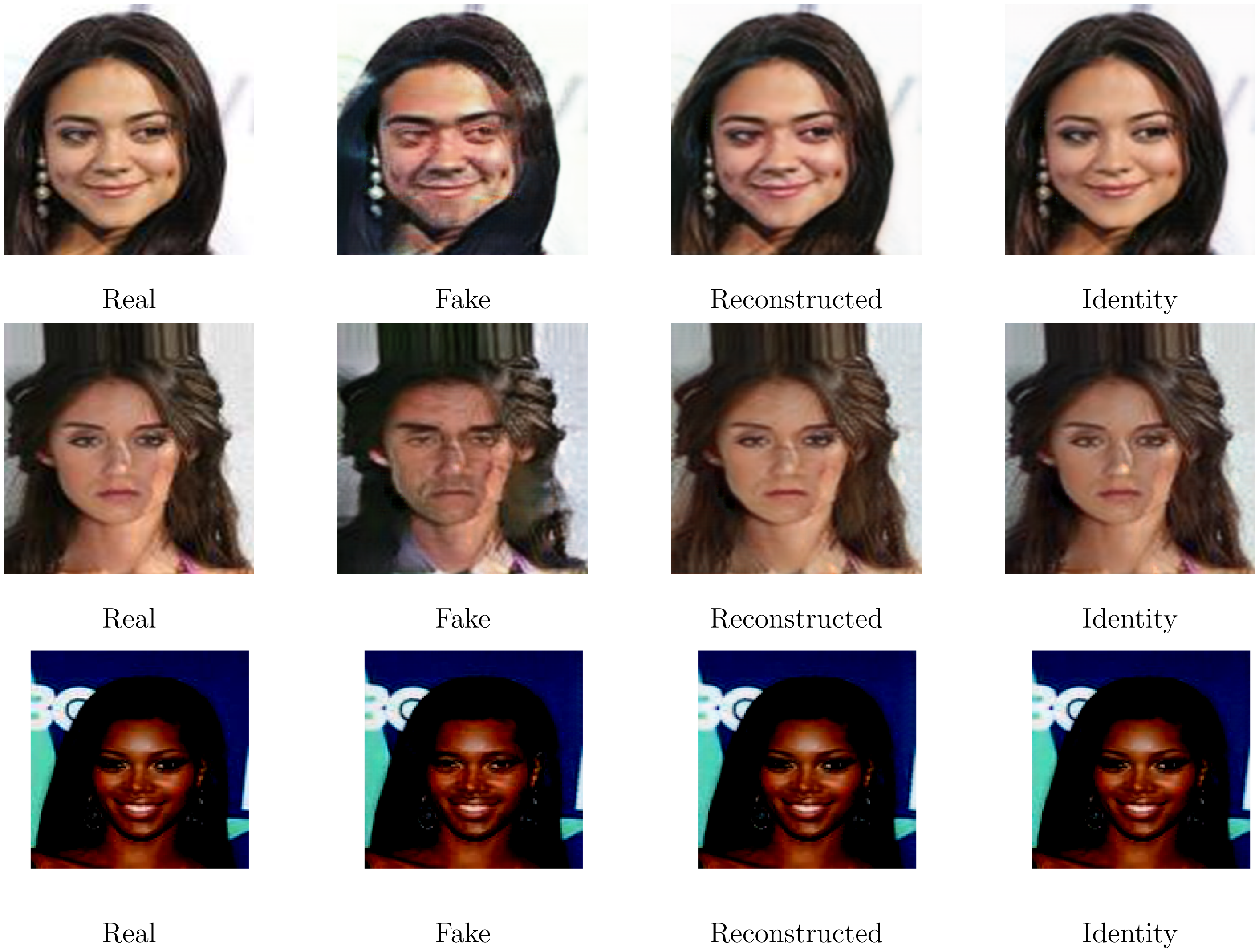


Figure 6.4: Female $\rightarrow$ Male

The generated fake images for the female to male conversion have a rougher skin texture and thicker lines around cheeks compared to the input images. This shows that the generator has made the faces more masculine without completely changing the original face. The reconstructed images are also very similar to the input images with very little changes to the facial features. Additionally, the images generated using identity loss are very similar to the input images.

6.3 Evaluation without Identity Loss

On running the model without implementing the identity loss, we obtained results that were not acceptable due to a change in the color composition of the fake images as seen below:

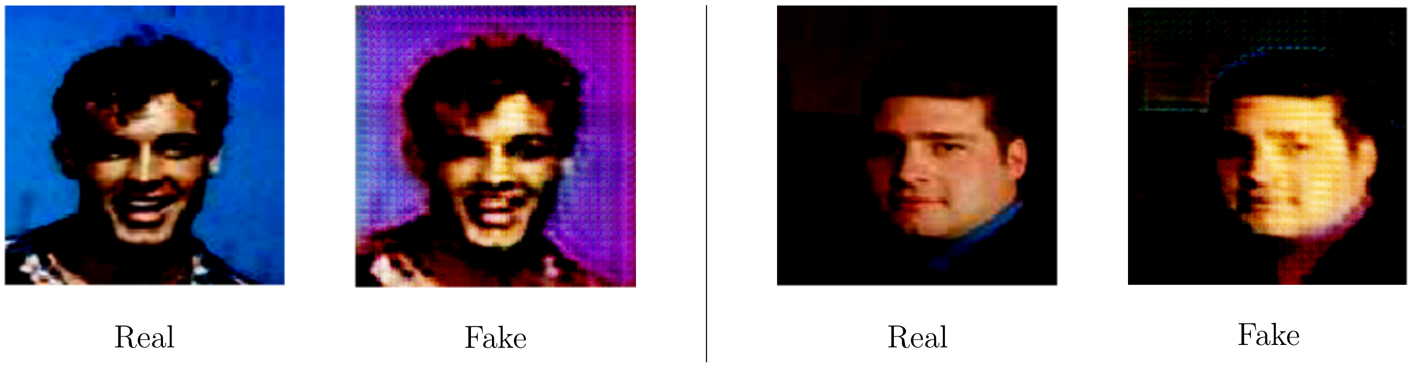


Figure 6.5: Generated output without identity loss

In the above figures, we can observe that the foreground and background color in the fake images have an unexpected change in color decomposition in contrast to the original images. This kind of modification is not desirable and hence we can can conclude that the identity loss is essential for our application to produce realistic faces.

6.4 Graphs

The generator and discriminator losses were plotted to evaluate the performance of the network. These plots were obtained from the tensorboard web application to which our losses were saved during the training.

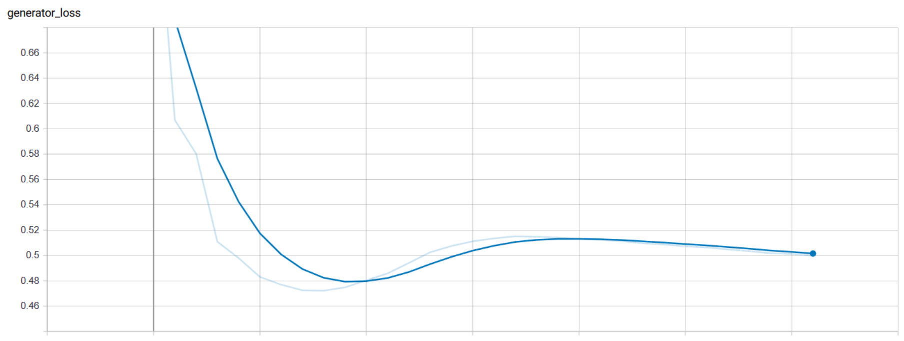


Figure 6.6: Generator Losses

The graphs indicated the min-max game between the generator and discriminator where a low generator loss corresponded to a high discriminator loss and vice-versa. This means that when a generator had a low loss, it generated very good fake images which the discriminator was not able to identify as a fake and hence the discriminator had a high loss.

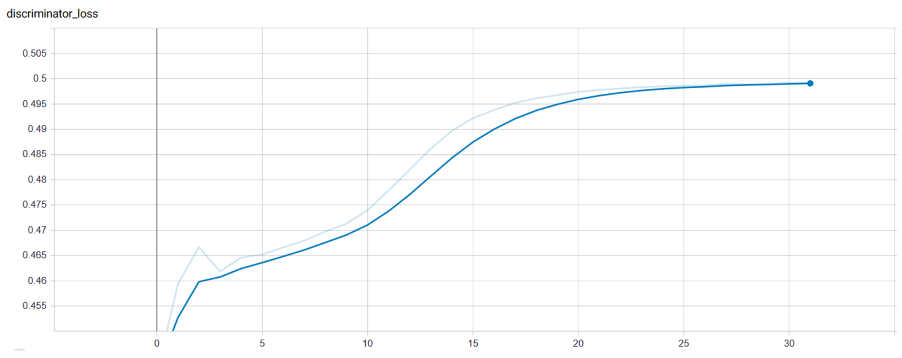


Figure 6.7: Discriminator Losses

As the training progresses with more epochs, we see that the generator loss gradually decreases and the discriminator loss increases. This is in line with our original objective where aims to maximize the discriminator loss and minimize the generator loss. Also, we notice that the generator loss stops decreasing significantly after a few epochs and hence we decide to stop the training at the end of 40 epochs.

6.5 Limitations and Discussion

Following were our observation during the implementation of the project:

- Implementing the ResNet-6 architecture resulted in fake images that had no significant transformation of the facial features, resulting in an almost identical image to the original image.

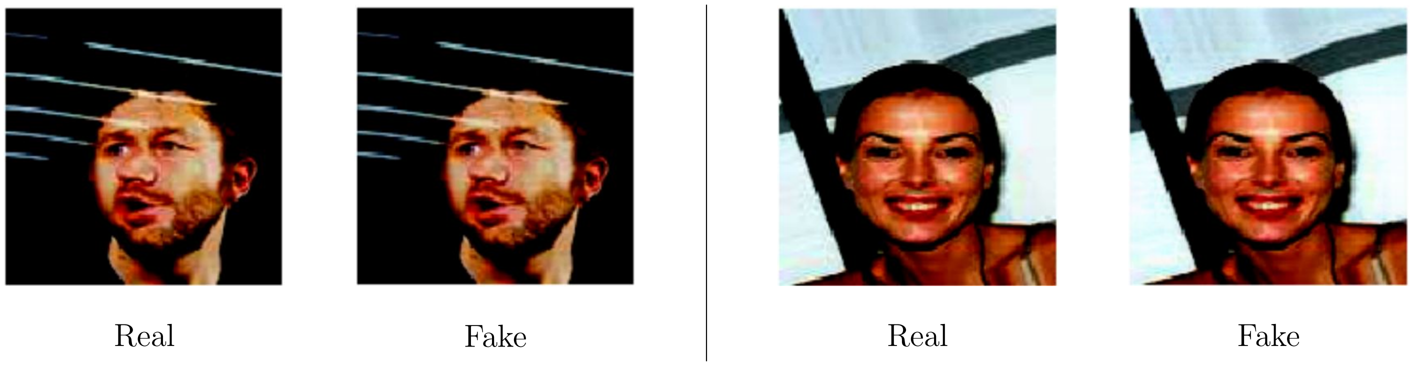


Figure 6.8: Output with ResNet-6

- From our experiments we see that the λ parameter for combining the various loss function has a significant impact on the results of the generated images. However, there is no clear criteria in setting this parameter which makes it difficult to tune it.

- Some of the generated images were grainy and blurry and hence not acceptable. The network also did not perform reasonable transfiguration effects on the hair.

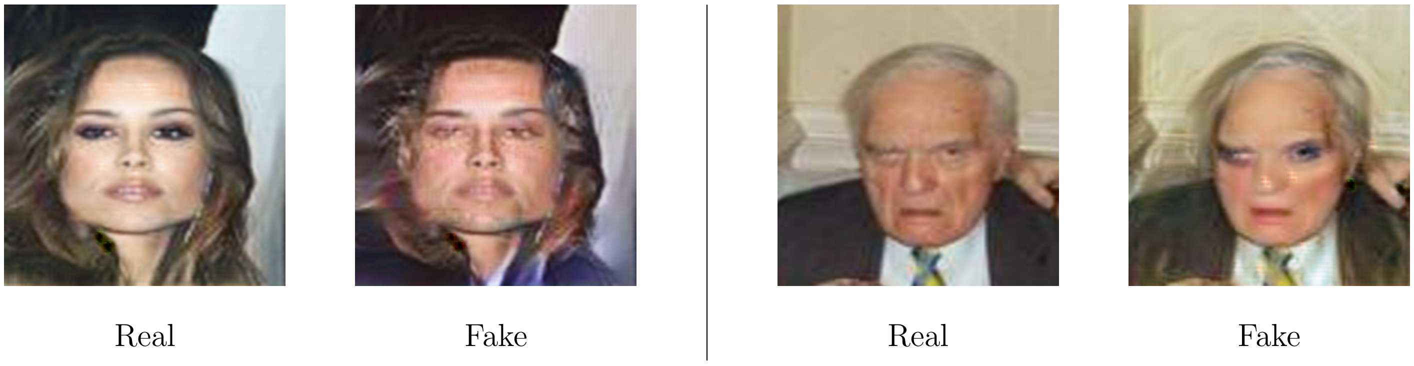


Figure 6.9: Bad quality

7 Future Work

- We can include a L1 loss on the CNN features extracted by the discriminator so that instead of focusing on the exact pixel to pixel mapping between the real and reconstructed image, our model can focus more on learning the general structures of the face, eye, nose etc [3].
- During the later stages of training, when information loss is necessary during the transformation stage, cycle consistency avoids the loss by enforcing a one-to-one pixel mapping. We can decay the weight parameter λ of the cycle consistency loss with each epoch so that it does not become a hindrance in the later stages of training while ensuring it remains greater than zero [3].

8 Conclusion

We have explored various concepts used in the CycleGAN architecture and studied the differences with other architectures used in domain to domain translation. By utilising the unique benefit offered by the CycleGAN structure, we successfully trained a model to do gender change on people's faces using our own keras implementation. The generated male and female faces were similar to the input images with a significant change in facial features. We have also experimented with identity loss in the training and conclude that it is necessary for good results.

9 References

[1] "Man disguised as woman used stolen credit card to buy french bulldog puppy from pet store: Police." Available at `https://abcnews.go.com/US/man-disguised-woman-stolen-credit-card-buy-french/story?id=59205833`.

[2] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[3] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *CoRR*, vol. abs/1703.10593, 2017.

[4] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.