Style Transformation on Human Faces using Feedforward and Generative Methods

Annie Gao
annie.gao@yale.edu
Yale University

Valerie Chen
v.chen@yale.edu
Yale University

Yichao Cheng
yichao.cheng@yale.edu
Yale University

ABSTRACT

Human faces are varied in nature and in styles, but often structurally similar. Recent approaches using feedforward and generative methods have demonstrated photo-realistic capabilities for artistic style transfers [3, 4, 8], which we believe hold potential for human facial feature transformation. We investigate the FastCNN, Cycle GAN, and Style GAN architectures to transform an input image to an 'aged' version of the image. We made various modifications including changing the loss network and loss weights, training data set, and adding a mapping network respectively to each architecture to handle the transformation of a new input image. The results for each architecture were varied. We present a qualitative analysis of each method's output transformation and example transformed images. Finally, we highlight some benefits and drawbacks of each method.

ACM Reference Format:
Annie Gao, Valerie Chen, and Yichao Cheng. 2019. Style Transformation on Human Faces using Feedforward and Generative Methods. In *Proceedings of ACM Conference (BIM Final Project).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The main goal of this project is to explore automatic transformation of features in human face images for the purposes of interactive applications and demonstrations. The project is motivated by a recent surge in popularity of apps such as FaceApp and Snapchat. These apps contain filters which are able to perform AI face editing to transform in real-time a current image of a person to an edited version, whether it be aging, changing genders, or enhancing smiles. Many online posts hypothesize that these apps use some version of generative adversarial architectures, so we are interested to see if we can build this proprietary technology ourselves.

Facial feature transformation is a subset of a larger class of work called style transfer. The goal of style transfer is to preserve the content of one image in the style of another by minimizing loss functions concerning content and style reconstruction, thereby preserving high-level features of both image classes. In this paper, we explore mechanisms to accomplish this transformation using both feedforward networks and generative adversarial architectures and gather qualitative survey results on the transformed results of each architecture.

2 RELATED WORK

The goal of neural style transfer is to use deep learning techniques to transform images of one style into another style. Fundamentally this is posed as an optimization problem to minimize the style and content distances between the two images. Prior work that relate to style transfer have used feedforward and generative adversarial networks. The three types of architectures that we will be investigating are [3, 4, 8]. Each method presents its own unique contributions towards style transformations, which we briefly highlight below.

The first method is a feedforward architecture, FastCNN [3], which introduced a pipeline using perceptual loss functions to perform artistic style transfer in real-time. One key contribution is their use of VGG16, pretrained on ImageNet, as a deep loss network that already contains perceptual and semantic information needed to compute content and style losses between output and target images. Using activation outputs from this loss network, they compute content and style losses from feature reconstruction outputs, which are used to train a feedforward image transformation network to preserve the content of the input image while transferring the artistic style of the target image in real-time.

The second architecture is the Cycle GAN [8], which simultaneously trains two sets of generative adversarial networks using unpaired training images from source domain X and target domain Y. The novelty of the approach lies in the introduction of a cycle consistency loss, which enforces that given input images from X, the resulting images should, after transformation first via one generator to Y and then via the other generator back to X, look similar to the original input images from X. This architecture is promising for style transfer, for the use of ResNet [2] in the generator can effectively extract and transform features, and the cycle consistency constraint can ensure that the main and definitive structure of a person's face remains unchanged while feature transformations alter the style. The freedom of not needing paired training data does not come for free, however, as implicitly, requirements for style consistency among images in each domain have become higher.

A third architecture that we will explore is the Style GAN [4]. Style GAN is the only architecture of the three that specifically presented results on faces. The novelty of the architecture is in the style-based generator to better control the image synthesis procedure, while keeping the rest of the set-up the same as prior work on generative adversarial networks. The style-based generator uses a multi-layer perceptron to transform a latent code to an intermediate latent space that can be adapted to different styles, so essentially they learn a mapping from a latent space to style transform rather than image directly to style transform. The main contribution of this architecture is the ability to mix style between two images from high-level aspects to coarse features. However, Style GAN is limited by its ability to do style transformation in that it can only transfers styles within the space of the latent dimension, so only generated images can be 'mixed'.

3 METHOD

We propose to explore three different recent architectures that have been demonstrated to be able to accomplish style transfer through preliminary work for the applications of facial feature transformation. For each method, we present a workflow to transform a given input image with respect to an attribute such as age or gender. We train each method on the CelebA dataset [5], which contains over 200K faces and 40 binary attribute annotations for each image, suitable for style transfer across a wide variety of facial features. In addition, Cycle GAN and Style GAN also used the IMDb-Wiki dataset [6] that contains 500k+ face images, where the numeric age labels allowed for the construction of more consistent training datasets where old and young are measured quantitatively.

3.1 Data Processing

From the CelebA dataset [5], we only extracted images that were marked as young, without glasses, and without hat, so as to remove partial facial occlusions. Then, in order to increase the chances of style transfer and decrease the effects of background clutter, we cropped the images to only contain a single face, as detected by a convolutional face detector network. We also resized all the face images to 256x256 pixels, normalized the pixel values, and loaded the resulting dataframes into separate training, validation and testing files for faster loading on subsequent runs. We preprocessed the IMDb-Wiki dataset [6] to only exact images that only contain a single face (via filtering with a threshold) and have valid date of birth and photo taken date. Then, we computed the age for the face in each image from date of birth and when the photo was taken to extract age <= 30 years old as young and age >= 60 years old as old.

3.2 Fast Style Transfer

One approach is to use two feedforward convolutional neural networks, an image transformation network and a pretrained deep convolutional model that serves as a loss network during training. Instead of using VGG16, an image classification network, we used VGGFace, a deep facial recognition network pretrained on the Labeled Faces in the Wild and Youtube Faces dataset, for an exploration of its impact on the content and style loss functions, which we will discuss later. We adapt the image transformation network architecture proposed by [3], and experiment with different hyperparameter configurations to test the effects on our output images. We are interested in investigating whether or not this transfer of artistic style can be applied to transferring stylistic attributes relating to facial features such as age.

Figure 1 shows the abstracted process of generating images. The input image also serves as the target content image, which along with the target style image, is fed through the selected activation layers of the loss network to produce target values for the loss functions. As images are generated by the image transformation network, they are also passed through the loss network to produce content and style losses, and the training process minimizes the weighted sum of all the computed losses.

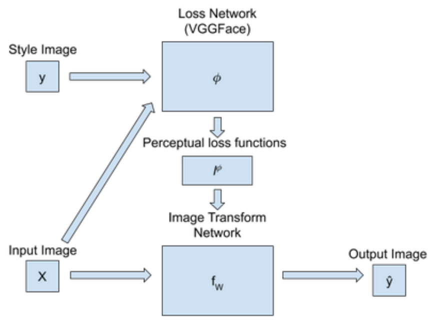

Figure 1: The proposed approach to using feedforward networks to perform style transform given an input image and a style image.

To generate loss metrics, we can extract activation outputs for specific layers J from the pretrained VGGFace model ϕ. As suggested by [3], we also used the activations from

$$J = \{relu1\_2, relu2\_2, relu3\_3, relu4\_3\}$$

to compute style reconstruction and feature reconstruction losses in order to train the image transformation network.

The feature reconstruction loss encourages the output image to be similar to the target image in terms of content and spatial structure, but because the distance is computed over feature representations instead of the original image pixels, it does not force the pictures to match exactly. The feature reconstruction loss [3] is computed as the squared, normalized Euclidean distance between the feature representations of the generated image $\hat{y}$ and the target image y in layer j of the VGGFace model:

$$l_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

The style reconstruction loss encourages the output image to be similar in style to the target image. It preserves stylistic features from the target image but not its spatial structure. To compute the style reconstruction loss, we first calculate the Gram matrices of the predicted and target images, which helps to measure the uncentered covariance of the features and thus provides insight on which features tend to activate together. The Gram matrix $G_j^{\phi}(x)$ of an image x uses the activations of the jth layer of the loss network ϕ, denoted $\phi_j(x)$. Let $\psi = \phi_j$. Then

$$G_j^{\phi}(x) = \frac{\psi\psi^T}{C_j H_j W_j}$$

and the style reconstruction loss is the squared Frobenius norm of the Gram matrices of the predicted and target images [3], computed as follows:

$$l_{style}^{\phi,j}(\hat{y}, y) = \|G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)\|$$

We now explain the choice of VGGFace over VGG16 as a loss network. For feature reconstruction and content loss, an image classification network is trained to preserve the classes of objects present in the content image. Our goal is the manipulation of facial features on human faces while the subject in the picture is still recognizable. Using image classification as a loss network could maintain a classification of "human face," but this could be too broad for recognition purposes. Instead, a facial recognition network such as VGGFace would help maintain the unique structure of each person's face as the style of individual features is changed. The other benefit of using a facial recognition network is that instead of transferring the artistic style of object classes found throughout the image, we focus on the style of tu features of the target image, thus also reducing the effect of background noise in the style target image.

Using the loss functions mentioned above, we train a convolutional neural network model to learn the image transformation network f_W, built according to the architecture summarized in the following table:

Layer	Activation Size
Inputs	(256, 256, 3)
32×9×9 conv, stride 1	(256, 256, 32)
64×3×3 conv, stride 2	(128, 128, 64)
128×3×3 conv, stride 2	(64, 64, 128)
Residual block, 128 filters	(64, 64, 128)
Residual block, 128 filters	(64, 64, 128)
Residual block, 128 filters	(64, 64, 128)
Residual block, 128 filters	(64, 64, 128)
Residual block, 128 filters	(64, 64, 128)
64×3×3 transposed conv, stride 2	(128, 128, 64)
32×3×3 transposed conv, stride 2	(256, 256, 32)
3×9×9 conv, stride 1	(256, 256, 3)

Table 1: Image transform network architecture

Following the architecture proposed in [3], each convolutional layer in the network is followed by batch normalization and relu activation. Each residual block comprises a 3x3 convolutional layer, followed by batch normalization, relu activation, another 3x3 convolutional layer, and another batch normalization layer. The downsampling in the network adds computational efficiency and increases the effective receptive fields of the convolution, and the use of transposed convolutional layers in the upsampling allows the upsampling function to be learned as the network is trained. The FastCNN pipeline is implemented in Keras using a Tensorflow backend.

3.3 Cycle GAN

There are three key aspects to the technical approach of Cycle GAN: network structure, loss function, and training data. We explain these in turn below.

The basic network structure is shown in Figure 2, where generator G generates fake Y images given input from X and discriminator D_Y scores for a given image how likely it is that the image is from Y. F and D_X are defined symmetrically. The two generator and discriminator pairs are implemented identically. The generator has three parts, which are the encoder consisting of 3 down-sampling layers that extract features, the transformer consisting of 9 residual blocks

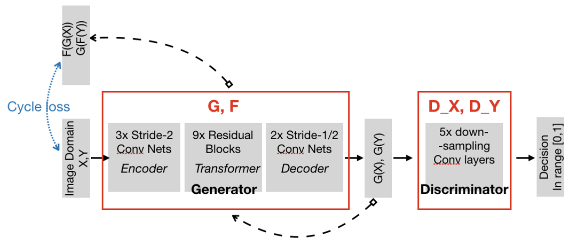

Figure 2: The proposed network structureimplementation for Cycle GAN

that learn feature transformation, and the decoder consisting of 2 fractionally-strided (aka. transpose convolutional) convolutional layers that perform inverse convolution operation to map features to pixels by filling in the details of a full image. The discriminator consists of 5 down-sampling convolutional layers that extract features unique to images in each domain. Additionally, techniques of instance normalization [7] and reflection padding are also used in the implementation.

The novelty of Cycle GAN resides in the idea of cycle consistency, where the two generators G and F are viewed as *inverses* of each other so that $G \circ F \equiv F \circ G \equiv I$, i.e., the distribution of images under map $G \circ F$ should be similar to the distribution of images in the domain Y, and similarly for the other direction as well. This perspective is incorporated in the full objective loss function, which is

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, X, Y) + \lambda \mathcal{L}_{cyc}(G, F)$$

where the first two terms are the standard GAN losses (including the identity loss and the adversarial loss) as in [1] and the third term enforces the cycle consistency constraint through measuring distance in the L1-norm as follows:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{x \sim p_{data}(y)}[||G(F(y)) - y||_1]$$

Here λ is a parameter used to adjust the relative importance of the cycle consistency loss, and the loss functions used in our implementation are L2-norm (the minimization of which is equivalent to minimizing mean square error) and L1-norm (the minimization of which is equivalent to minimizing mean absolute error).

Then during training, in order to update the generators via both the adversarial loss and the cycle consistency loss, we define two composite models, $G \longrightarrow D_Y \longrightarrow F$, and $F \longrightarrow D_X \longrightarrow G$. In the composite model $G \longrightarrow D_Y \longrightarrow F$, generator G is updated via four losses: the adversarial loss $||1 - D(G(X))||_2$ (L2 norm), the identity loss $||Y - G(Y)||_1$ (L1 norm), the cycle consistency loss in the forward direction $||X - F(G(X))||_1$ (L1 norm), and the cycle consistency loss in the backward direction $||Y - G(F(Y))||_1$ (L1 norm). Similarly, generator F is updated via the composite model $F \longrightarrow D_X \longrightarrow G$. Note here that for simplicity in notations, we used X and Y to mean the batch of images drawn from the distribution of images in each domain respectively, as in the above $\mathcal{L}_{cyc}$ definition.

The final key aspect is training. After processing the datasets as detailed in 3.1, we trained four kinds of models. The first kind was trained using the young and old datasets (male and female) from the CelebA dataset, with relative loss weights for the four losses (adversarial loss, identity loss, cycle consistency loss in the forward direction, cycle consistency loss in the backward direction) being [1, 5, 10, 10]; the second kind was trained using the young and old datasets (male only) from the IMDb-Wiki dataset, with relative loss weights being [1, 5, 10, 10]; the third kind was trained using the young and old datasets (male only) from the IMDb-Wiki dataset, with relative loss weights being [5, 1, 10, 10]; the forth kind was trained using the young and old datasets (male only) from the IMDb-Wiki dataset, with relative loss weights being [5, 5, 10, 10]. Note that these loss weights are relative values that will then be normalized in the Tensorflow backend implementation. The first model was trained on a total iteration of 20,000+ images, and each of the other three models was trained on a total iteration of 10,000+ images.

3.4 Style GAN

Style GAN [4] allows for control over style features of an image through manipulation of the latent code. Each image that is generated by the Style GAN generator is associated with a source latent code. Style GAN mixes the 'styles' of two different images, which is the intermediate latent code, by injecting an affine transformation of the two different styles at different layers of the upsampling process. Specifically, 'style' is fed into the adaptive instance normalization (AdaIN) modules after each convolution layer in the synthesis network. The AdaIN operation scales and biases the image feature x_i towards style $y = (y_s, y_b)$ using the following:

$$AdaIN(x_i, y) = y_{s,i}\frac{x_i - \mu(x_i)}{\sigma(x_i)} - y_{b,i}$$

We propose to utilize this mixing functionality for the purpose of style transformations such as changing the age or gender of face by leveraging the control of the latent code which housed in the generator network. The set-up of the Style GAN follows that presented in [4].

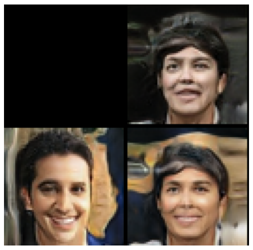
Figure 3: Example generated and mixed images from our own trained Style GAN. The image at the bottom right corner is an example of mixing the two generated images on the top and left.

To achieve style mixing, the model should be trained on data which captures variation in the attribute we are interested in. For example, when doing style transformation on age, a variety of ages should be represented in the training data that the Style GAN encounters throughout the training process. Figure 3 demonstrates through an example style mixing using our own trained Style GAN network on an image size of 64x64. We found that a limitation of training our own network is that we would spend all limited computational power on basic training, which does not have the capability to do style transfer on an input image. To progressively scale up to the final size of 1024x1024, the original Style GAN paper mentions that it would take over one month of training on a single GPU. Thus, for the rest of this paper, we consider a pre-trained Style GAN model on 256x256 images with a latent dimension of 512.

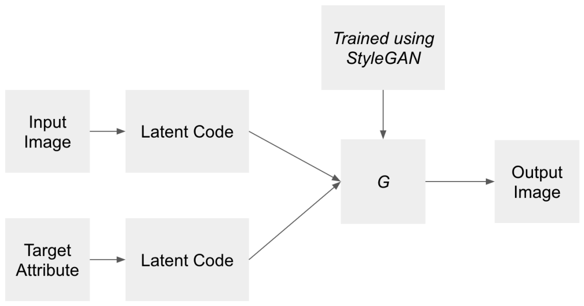

Figure 4: The proposed approach to utilize Style GAN for image feature transformation. The key contribution is not the training of Style GAN but mapping of a new, real image to the latent space of the generator.

Given a new input image, Figure 6 presents a high-level overview of the overall pipeline using the pre-trained Style GAN model. The input image and target attribute are provided by the user. In this project, we propose a perceptual loss method to find an appropriate latent code for the new input image, which allows us to then mix image styles using the pre-trained generator. We pose this perceptual loss as an optimization problem:

$$\min_t MSE(\phi(s), \phi(G(t)))$$

where s is the input image, t the predicted latent code, $G(x)$ the generated image given latent code x, and $\phi(x)$ maps an image to visual features from VGG16. Specifically the layers of VGG we extracted to make up the function $\phi(x)$ are the 3rd, 8th, 15th, and 22nd which correspond to the ReLU outputs. The pipeline for the perceptual loss method is as follows:

(1) Guess latent code
(2) Use generator to generate image given latent code
(3) Extract image features using a forward pass through $\phi(x)$
(4) Calculate MSE Loss
(5) Backpropagate loss through to input latent code, holding all other parameters fixed
(6) Repeat from step (2) for fixed number of steps or until error sufficiently small

This training process was optimized using the Adam optimizer with a learning rate of 1 and a criterion of standard mean squared error. All code for Style GAN was implemented in Pytorch. Each new input image would have to be processed in such a way to find the corresponding latent code in the latent space of the generator.

We consider a fairly simple way to mix the latent code of the input image with one satisfying the target attribute. For example, if we are interesting in 'aging' the input image, we should select an image of an older person to do style-mixing on. One method to select such an image is to manually record the latent code of an image that one would consider 'old'. Another method could be to use an age detection network on many generated images and select the latent code of an image that has an age above a certain threshold. For simplicity, we used the first method for this project. At each upsampling level, the generator randomly selects from the set of available styles to inject into the AdaIN module.

3.5 Evaluation

To analyze the training process, we compare learning curves from each of the different network architectures to assess training process stability. While this metric does not directly inform us about image quality, it will be interesting to observe the difference between training a feedforward network and generative adversarial network.

To analyze the transformed images, we evaluate the results through visual analysis for how realistic the image is. We surveyed other students with a few questions that help us understand in what cases each architecture does better and identify potential directions for improvement. Questions we asked include:

(1) Given these images transformed by each network, rate each on a scale of 1 to 5 how realistic the face is
(2) Given original and transformed pictures, guess the age of the people pictured. We can then measure the difference in estimated ages and see how effective our method is for various age groups.

4 RESULTS

4.1 Fast Style Transfer

To evaluate the output produced by each network, we used three unaltered content images and one unaltered style image, as shown in row 1 of Figure 5.

As a baseline, we trained an unmodified implementation of the feedforward network presented in [3], with the results in row 2 of Figure 5. As expected, the style transfer seems to have been applied uniformly to the input images, with "wrinkles" transferred everywhere in the faces without any particular localization or direction. It also did not significantly change the content of the images. As a sanity check, we also trained a model on a smaller learning rate of 0.0001 instead of 0.001, and a model with a greater style weight of 1.5 instead of 0.5. An additional epoch was added to train the model with lower learning rate, since after two epochs, the training loss had not appeared to stabilize. As shown in rows 3 and 4 of Figure 5, this did not improve our results, since the pictures also do not modify the facial structure of the subjects, and instead just blanket the pictures in white, which we hypothesize to be from the light illuminating the face of our original target style image. Note that the reconstruction of the target image in row 4 lacks content and style precision, hinting that simply adjusting content and style weights would not be productive. With hyperparameter tuning on a network trained with a VGG16-based loss largely ruled out, we trained another network that used VGGFace as its loss network. The results are shown in row 5 of Figure 5. Although the edges of the picture are noisy, this set of results is the only one so far with noticeable change in the structure of individual facial features. We think the lower image quality of the pictures may be due to training issues from the ethnic discrepancies between the content and style images, as well as the biased demographics present in our dataset. To address this, we trained another model using the target style image shown in row 1 of Figure 6, which produced the results in row 2 of Figure 6. Here, the identity reconstruction of the target image appears more convincing than that of row 5 in Figure 5, and the generated output images are also less noisy. The illumination in the target image's face also transfers to specific, matching features on the input images, although the hairline, face shape, and color scheme are also forced upon the input images, which detracts from their realism.

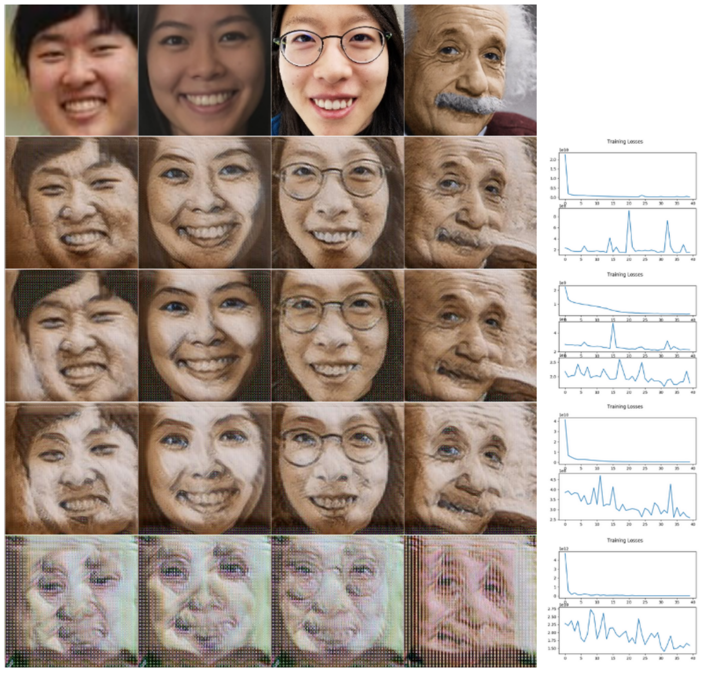

Figure 5: Row 1: 3 original content images, target image. Row 2: Unmodified architecture. Row 3: Smaller learning rate. Row 4: More style weight. Row 5: Modified loss network (VGGFace). Rows 2-5 also have a graph of training losses over epoch steps.

4.2 Cycle GAN

Overall, the trained Cycle GAN models did not produce very promising results, though we made some interesting discoveries when comparing results from the four models that were trained differently. From Figure 7, we see that the fake old images generated by the first model are almost replicas of the original input images, except for some color variations, and there was no sign of aging. The images generated by the second model also did not have obvious

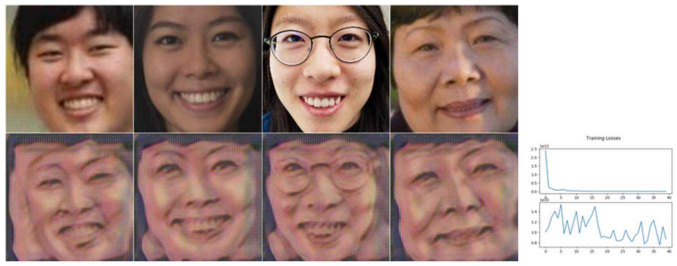

Figure 6: Row 1: 3 original content images, target image. Row 2: Output of first row images from network trained with VGGFace loss, along with graph of training losses over epoch steps.

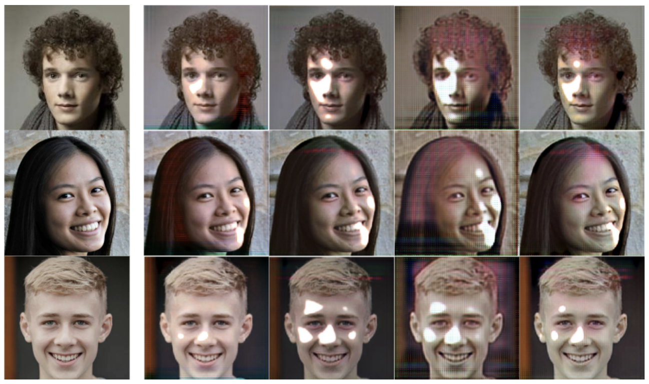

Figure 7: The first column has the original images. Column 2:5 correspond to the fake old images generated by the best model trained in each of the four ways respectively.

signs of aging, but there was a greater level of image transformation. The images generated by the third model are the most coarse of all, as shown by the large pixelated regions displaying highly exposed colors. The images in the last column generated by the forth model, however, showed some promising signs. Most parts of each image are reconstructed well despite some pixelated spots, and compared with the original images, these faces look a little older even just from the overall color scheme. We also plotted the loss and accuracy graphs for each of these models, which all shared the general trend of those shown in Figure 8.

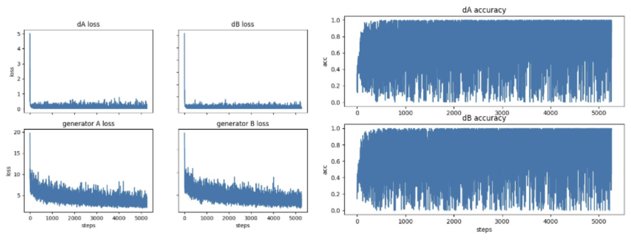

Figure 8: Loss and accuracy graphs plotted using data obtained from training the first model.

From the accuracy plots, we see that the models did not converge, and yet from the loss plots, there was a seemingly promising improvement. We suspect that mode collapsing might have happened, where with the high relative weight on the two cycle consistency losses, the generators learned the easiest way to minimize that loss, which is to output a near-identity map. This prompted us to experiment with changing the relative loss weights as we did in training the third and forth model.

The results in Figure 7 to a large extent corroborated with our guess, although we will need more systematic experiments to further confirm it. By decreasing the relative weights of the cycle consistency losses when training the forth model, we saw a noticeable increase in image transformation, and there was also a tiny sign of aging. By decreasing the relative weight of the adversarial loss and increasing the relative weight of the identity loss when training the third model, we saw the greatest level of image transformation from the results, given that the last three models were all trained under a similar number of training samples. Additionally, we also saw that the underlying color schemes of images in the last two columns are similar. There is reason to believe that with more training, these two models will produce more realistic aging results.

On the other hand, changing the training dataset from the first model to the second model did not see a noticeable improvement in performance. This could be due to the lack of consistency in both training sets, where images in each domain have completely different backgrounds, colors, etc. Overall, besides insufficient parameter and model tuning, we think the failure of Cycle GAN applied to style transformation was to a large extent due to the highly noisy training data, whose lack of a uniform style made it very hard for the GAN networks to learn the style transformation at the facial feature level.

4.3 Style GAN

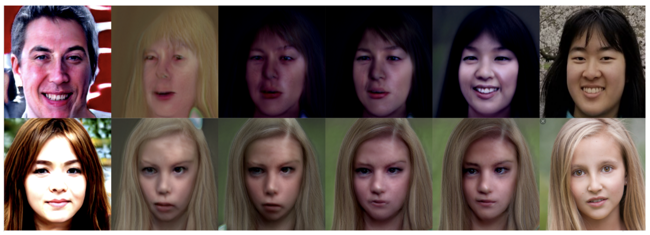

Figure 9: The top and bottom row correspond to two example evolution over multiple iterations of optimized faces. (Left-most) Image generated by initial latent code guess. (Right-most) Target image to match. The second image from the right is the final optimized image.

Overall, the perceptual loss optimization process was fairly successful in finding a latent code whose generated image was perceptually similar to the target image. As shown in Figure 9, even if the initial latent code guess corresponded to an image of the opposite gender, the eventual optimized image would look like the correct gender. In general the optimized photo would capture the correct face shape, hair color, and whether or not the real photo had a smile. However, it is lacking in the ability to mimic some finer grain details. For example in the top row of Figure 9, the optimized photo lost some of the definition in the hair and eyes and the bottom row was unable to match the exact face expression made. Perhaps this is because the faces could be more closely aligned with the training images in terms of cropping or because the size of image we are using is 256x256 which would not capture as granular details as a larger 1024x1024 image.

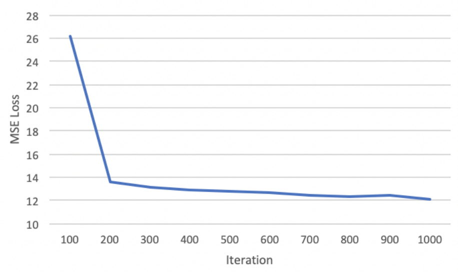

Figure 10: Loss curve for perceptual similarity optimizing. After about 200 iterations, the loss reaches a (local) minimum. Perhaps with additional hyperparameter tuning we would achieve better results.

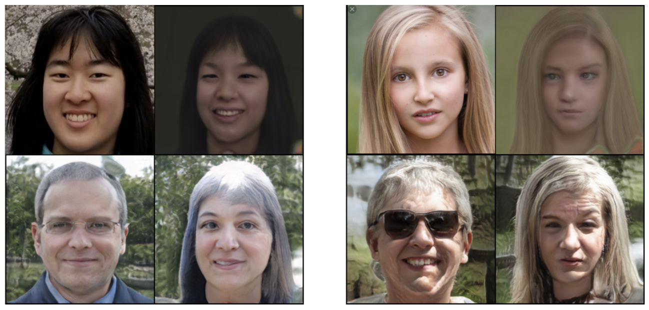

Figure 11: For each set of four images: the top left image is the input image, the top right image is the generated image, the bottom left is the selected target attribute image, and the bottom right is the mixed image.

Figure 11 demonstrates two examples of faces after performing style mixing. We found that there is substantial variability in terms of the quality and how realistic the generated face outputted from style mixing were. In some cases, more features from the original image is preserved that in other cases. This might be due to some underlying compatibility between the two latent codes, which is difficult to disentangle in the generator. An additional factor could be the variability in goodness of perceptual reconstruction of the input image, which will have downstream effects on the style-mixing.

4.4 Qualitative Comparison

We conducted a survey among 17 participants, consisting of two sets of original and modified images for each of the three architectures. Each set of images was accompanied by the following questions:

(1) Guess the age of the original face?
(2) On a scale of 1 to 5 (1: not realistic at all, 5: very realistic), how realistic is the modified face?
(3) Guess the age of the modified face?

We dropped one extreme outlier response fully from the evaluation and analyzed the responses from remaining 16 people. For each of the three network architectures, we calculated the average of the difference between the perceived age of the modified face and the perceived age of the original face. We also calculated the average realism rating, with the results shown in Table 2. We found that Style GAN had the highest average increase in perceived age. Given that Style GAN had been designed for the purpose of mixing faces, this result was in line with our expectations. Additionally, Cycle GAN had the lowest average increase in perceived age. This was expected because we were not able to achieve age transfer with the architecture so there would not be a noticeable difference between age difference. Finally, FastCNN had the lowest average realism rating. This is perhaps because FastCNN was primarily designed for artistic style transfer and so the output image was not optimized for realism.

Architecture	Average Increase in Perceived Age	Average Realism Rating (1-5)
FastCNN	20.196875	1.65625
Cycle GAN	0.65625	3.5625
Style GAN	30.9578125	2.78125

Table 2: Evaluation Results comparing Style Transfer Architectures

5 CONCLUSION AND FUTURE WORK

Overall, we found that the main benefits of feedforward methods such as FastCNN was the speed and relative ease to train. Generative architectures, Cycle GAN and Style GAN, required substantially more computational time and power. Due to these limitations on generative architectures, we found mixed results in terms of the aging capabilities. However, generative methods outperform feedforward methods in terms of their power to generate a variety of images. This provides Style GAN with the capability to mix styles of images easily and Cycle GAN to learn transformations between unpaired images. Generators have latent spaces from which they sample new images over a diverse space while the feedforward method only accepts one target style per trained network. Specifically, FastCNN tries to fit all input images to a very specific structure, which leads to less realistic outputs. In the future, results on each architecture might be improved if we improve the architecture's capability to disentangle style features. This might be done on the model perspective or the training data end. Additionally, since our goal is to build interactive applications, we might want to incorporate a metric for photorealistic outputs to encourage the model to produce more convincing outcomes.

REFERENCES

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[3] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*. Springer, 694–711.

[4] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4401–4410.

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

[6] Rasmus Rothe, Radu Timofte, and Luc Van Gool. 2015. DEX: Deep EXpectation of apparent age from a single image. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*.

[7] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022* (2016).

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.