Stylegan Encoder Interface

Stylegan is an image synthesizer than uses generative adversarial networks (GANs) to transfer style from one image in the stylegan latent space to another image in that space. This paradigm diverges from that of traditional GANs, particularly in the inputs supplied to each. That is, whereas traditional GANs take an image as input and train from that, stylegan takes as input a latent space vector representation of the image. This latent vector represents features of an image at 9 different level of abstraction. In this vector form, the features of two images can be easily swapped, producing a latent representation that includes characteristic of both of the original images. This vector can then be passed through stylegan's synthesis network to output a human-interpretable RGB image.

However, as interesting as this sounds, **stylegan is only designed to work with images that were a part of its original training set**. In this sense, the original stylegan implementation is somewhat of a closed system; images not in the training set have no latent space representation and therefore cannot be style-mixed with other images.

By itself, the lack of transferability of stylegan makes it more of an experimental concept tool rather than an application that has utility in the real world. Because of this issue, I sought to attempt to find a way of representing real, human-captured photos as feature vectors in the latent space of stylegan. After a bit of research, I found that **the most practical way of making such a transformation was by training a latent vector on a real photo**. This training is done by forward propagating some (intelligently initialized) latent vector through stylegan's synthesis network, comparing the output image to the original image, and updating the latent vector based on the loss between the two images. Eventually, the gradient descent will converge to an input that generates an output image almost identical to the original.

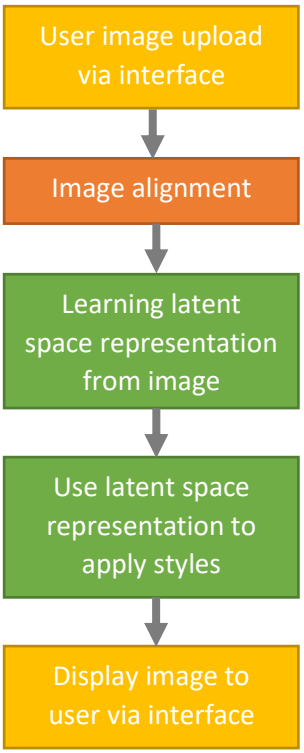

Figure 1: Flow of control for user interface. Yellow blocks represent components of the UI, green squared represent components of the stylegan network, and the orange square represents an image alignment system separate from the other systems.

Initially, I had attempted to implement such an optimization system on my own. However, I eventually ran into several issues while working with TensorFlow and so, given the time constraints, decided to take an alternative approach. Specifically, I found a project that was also working on this issue and I decided to extend this project by adding a user interface. In addition to being more successful in finding a solution to this problem, this project also differed from my implementation in that instead of mean-squared error loss, it used a perceptual loss function call VGG-16, which is specifically designed for GAN traning. This project also implemented an automatic image aligner, for aligning user non-conforming images so as to be in the correct format for training.

The user interface I implemented has a very basic look to it but is nonetheless performing complex operations in the backend. The flow is as follows: (1) The user uploads their own photo to the website. (2) A preprocessing system properly aligns the image for training. (3) The synthesis network of stylegan is used to train a latent vector, during which the aligned image is used as a reference for calculating loss to iteratively update the

latent vector. (4) The resulting latent space image is saved, and the resulting latent space vector is "style-mixed" with other images already existing in the latent space of stylegan. (5) These style-mixed images, the existing latent space images, and the learned latent space image are all combined as an image array and displayed on the user interface. This process is shown in Figure 1. The interface itself is shown in Figure 2.

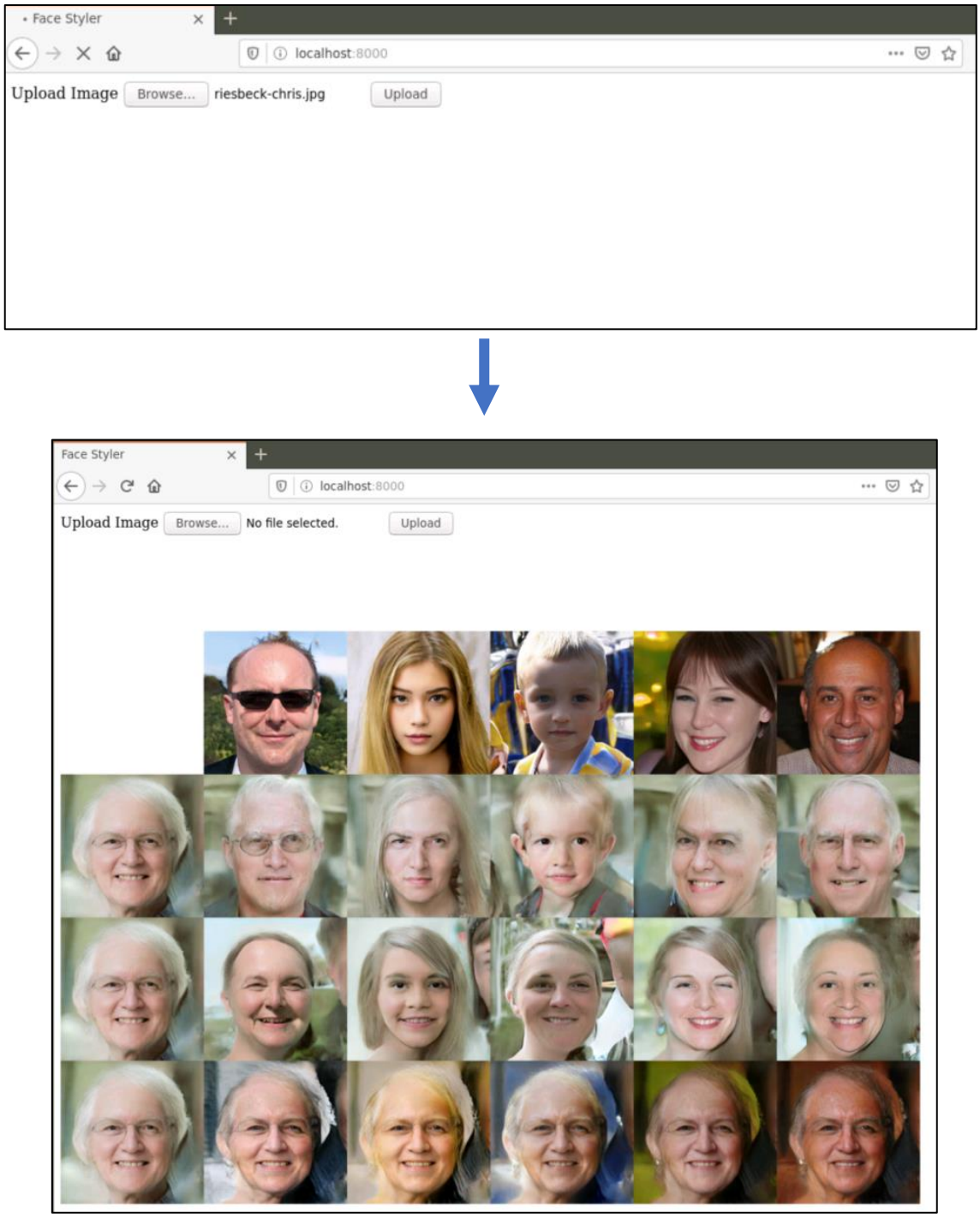

Figure 2: Example usage of stylegan encoder web interface.

Future work on this project would be focused in improving training and improving the UI. Specifically, training might be improved by modifying the loss function (perhaps to a combination of mean-squared error and VGG-16) used for learning the latent space representation of the image. The user interface might also be improved upon my adding more functionality (e.g. combining to uploaded photos, select different styles to apply, select style transfer granularity, etc.) and by simply making the site more aesthetically pleasing.