Refining Style Image Inputs in Neural Style Transfer

Vadim Kudlay
vadim.kudlay@richmond.edu

Victor Chen
victor.chen@richmond.edu

May 2, 2020

Abstract

In this project, we attempted to refine style image inputs for neural style transfer between anime characters and real human portraits. We noticed that style transfer quality depended greatly on the quality of the provided dataset and we intended to clean up style image properties to match those of the content image. We hypothesized that doing so would improve the resulting stylized images, since the inputs were more similar. We used a convolutional neural network (CNN) to implement an arbitrary style transfer and used a variety of machine learning algorithms in order to refine and augment the style image. Using these techniques we were able to minimally improve the quality of the style transfer that we performed since much of the excess noise in the background was removed.

1 Introduction

Much of machine learning relies on a myriad of data to work with. Sometimes, that data cannot be applied directly for the purposes that researchers are interested in observing. This is specifically an issue we would like to address in neutral style transfer. Neural Style Transfer is an optimization technique that uses two images, a content image and a style image. The content image is the image that is taking on or assimilating the style of the style image. The resulting image is a stylized image that is fairly interesting to analyze, both for artistic and academic purposes. This technique is commonly used with artworks to emulate an artists' style. We specifically were interested in using style transfer on human faces to make them look more "anime-like", or at least take on certain color palettes and facial features of animated characters. During our research in neural style transfer, we noticed that some stylized images were lacking in quality and sharpness and wondered if they could be refined. We hypothesized that matching certain properties between the two images, such as size, shape, and color, could increase the quality of the style transfer. Such research could be scaled up to be used more conventional purposes of style transfer, as the lack of refined data is a major obstacle when it comes to machine learning.

Beyond the amusing nature of style transfer, this project has significant academic value for us. This task required us to learn more about deep learning, specifically convolutional neural networks (CNNs), which was both a challenging and fun experience.

2 Survey of Related Work

1. **Google Deep Dream Generator**

   Google's gallery of deep dream pieces is one of the most popular and influential projects with regard for style transfer. On the website, they feature a myriad of style

transfer examples as well as tools that could be used to generate them as an artist or other practitioner. While pieces like the stylized Starry Night piece by Van Gogh may be seen as unsettling and confusing, this work shows the subjective limits of style transfer and how we could possibly optimize image inputs for less abstract results. The projects uses three tools/techniques: deep style, thin style, and deep dream, each of which have their own benefits and uses. This work is relevant to ours but different in that our focus lies in refining the input rather than the style transfer algorithm itself. (*Google Deep Dream Generator*, 2020)

2. **Style Transfer with GANs on HD Images**

   Style transfer is generally performed using a generative adversarial network (GAN) architecture. Going beyond style transfer, they can be used to generate convincing input-output mappings between general data distributions, be they text, images, sound waves, etc. These operate by defining two neural network components, the *generator* and the *discriminator*, to compete against each other where the former maps input to an output and the latter analyzes the output to try to identify if it belongs to the class of real output. By setting up this zero-sum game, such a system is able to mutate inputs like video feeds and raw images into vehicular decisions and stylized equivalents (respectively). The main issue with GANs is that training such models is computationally expensive, and was not reasonable for a long time. With the rise of powerful GPUs, however, the processes can be parallelized on modern processors and mathematical computations can be streamlined to make the time at least reasonable to train. We choose to forgo GANs for CNNs, or convolutional neural networks, and decided to work with pre-trained CNN models offered by Tensorflow through their Tensorflow hub. (Pasini, 2019)

3. **Generate Anime Style Face Using DCGAN and Explore Its Latent Feature Representation**

   This research shared our interest to generating anime style attributes but took a slightly different approach in that we wanted to transfer the style of pre-existing style images to the content image. This research generated anime style faces from scratch by training a type of CNN called a deep convolutional generative adversarial network (DCGAN), which is a GAN structure with many convolutional layers. Although the technology was impressive, the results were fairly different than what we were aiming to emulate. (Wibowo, 2019)

4. **User-Guided Deep Anime Line Art Colorization with Conditional Adversarial Networks**

   This research does not focus on style transfer in the traditional sense, but considers the complexity of generating certain style properties. It used a GAN architecture to colorlize cartoon/anime content with a mask of stroke ”hints” from the user that are collected through an interactive application. Although our project was not directed

at colorization, augmenting visual properties such as color was in the general area of things we looked into for inspiration. (Yuanzheng, Xinzhu, Zhihui, Haojie, & Zhongxuan, 2018)

3 Formulation

We formulated our problem as mainly a classification problem. This is because we will need to classify facial features for the style image in order to refine it. We also considered the backgrounds and overall size of the image in comparison to the content image. We detected facial features using LBP Cascades, a machine learning object detection algorithm. and used semantic segmentation for background detection/removal. Semantic segmentation models aim to assign semantic labels to multiple objects within an image, such as: tree, sky, person, and bird. We specifically used a CNN, in order to perform the style transfer. A convolutional neural network is a class of deep neural networks used specifically for analyzing images. In the most basic sense, the network breaks down image data into features and uses this information to classify the original image. Often these networks are trained on large amounts of labeled data to increase the accuracy of the network. The CNN that we are using is found in an style transfer model in the Tensorflow Hub. The model, "arbitrary-image-stylization-v1-256", is pre-trained so we can reduce excess complexity in our code base. Furthermore, the model operates with a fast style transfer algorithm, which allowed us to save time when performing experiments. Since our project is focused on analyzing and isolating visual imagery, we decided that convolutional neural networks were the right choice. Additionally, the Tensorflow tutorial on style transfer utilized VGG19, a 19-layer deep convolutional neural network which we found to be extremely effective in its demonstration.

4 Approaches

We used Python3 for this deep learning project and utilized a variety of libraries and public GitHub repositories for specific function implementations. We used Keras and Tensorflow for working with the convolutional neural network. As stated before, we reduced our complexity by leveraging a preexisting style transfer model found in the Tensorflow Hub after testing with the VGG19 implementation initially. One of the GitHub repositories that we augmented to suit the experiments was lbpcascade_animeface (nagadomi, 2011), a LBP Cascades implementation of face detection trained specifically for anime that utilized OpenCV, a library used for real-time computer vision. Another repository that we used was image-background-removal (susheelsk, 2018), which used Tensorflow to perform portrait segmentation.

We also used matplotlib and the Python Image Library (PIL) to visualize experimental results and plot our image figures. In order to adjust our images, we used the Numpy package for cropping/slicing with a bounding box and the Tensorflow image module for

resizing. Our chosen dataset was the aggregated dataset provided by Anime-Face-Dataset (Mckinsey666, 2019), which supplied us with a myriad of pre-cropped portraits of anime characters scrapped from miscellaneous sources.

5 Experiments and Analysis

For experimentation, we focused on several different augmentations for the style image. These were: background-removed, face-detected, cropped, and averaged.

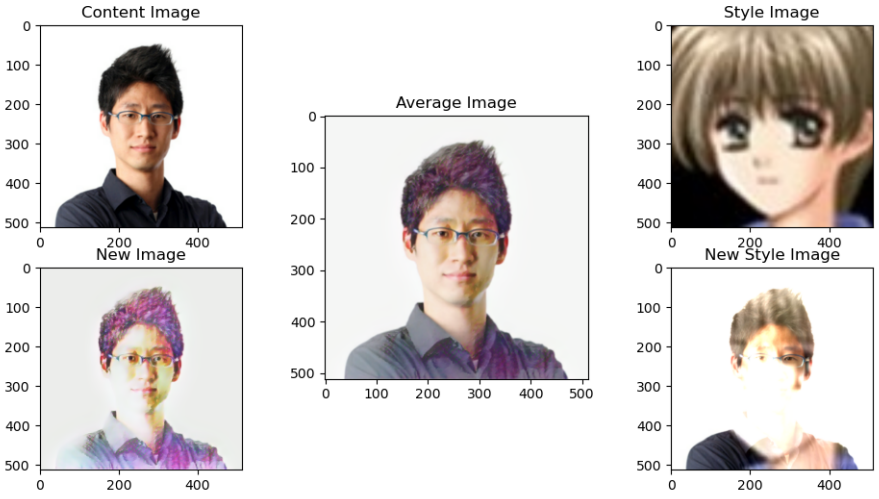

Figure 1: No changes to style image

In figure 1, we see the original style image and the content image We can see that the ”new image” or stylized image is distorted in coloration, to account for this extremity, we took the average of the stylized image and content image. Take note of how the style image is not aligned correctly with the content image. Part of our hypothesis is to crop that style image and see if that changes the resulting stylized image. We can also change the average to take into account the style image as well.

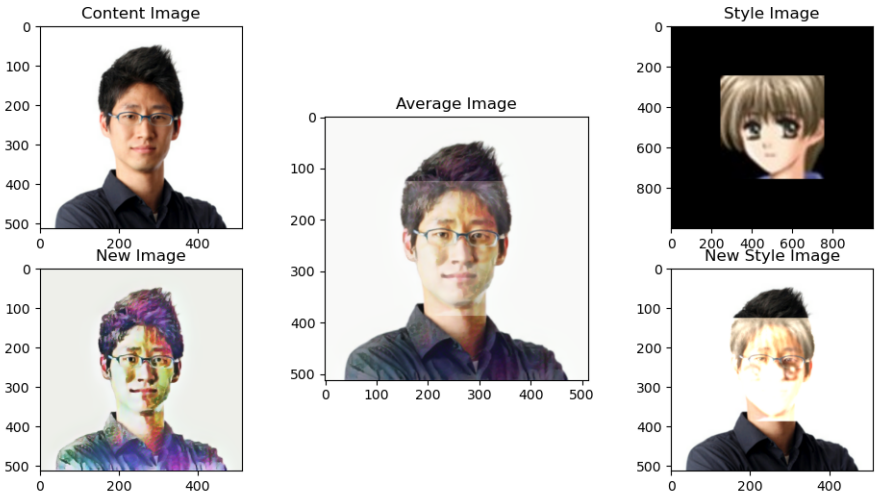

Figure 2: Average of 3 and cropped

We also augmented the style image by removing the background. The library that we used to implement portrait segmentation was not perfect around the edges but was sufficient in clearing out a majority of the background.

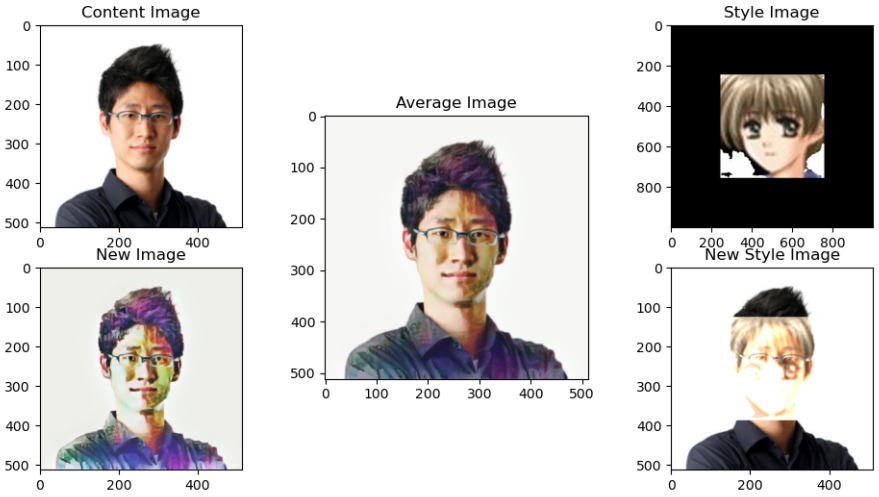

Figure 3: Removed background and cropped

Take note of how the stylized image distorts significantly from the cropping. The average image also attempts to reduce to noise that resulted from the style transfer.

For the facial detection algorithm, we will show two different style images to see how to bounding box is determined (see figure 4).

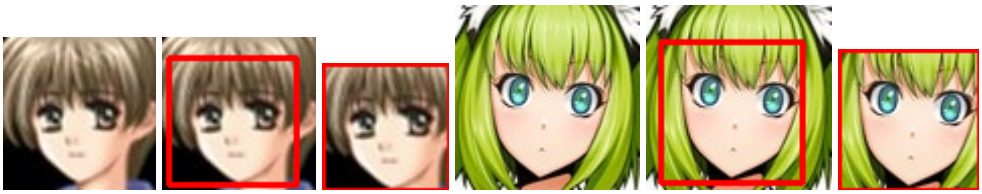

Figure 4: Bounding box from face detection

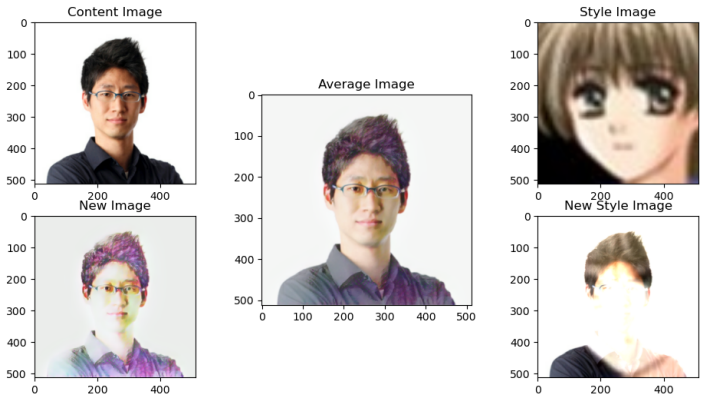

Figure 5: Face detected without cropping

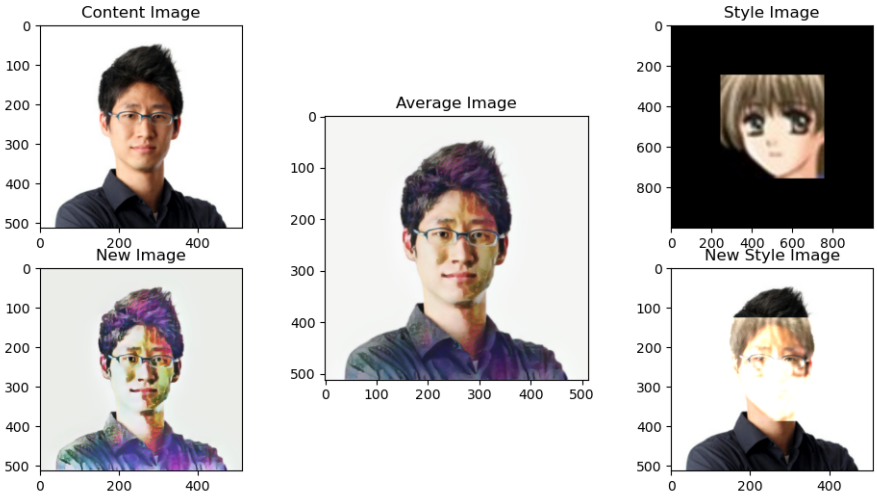

Figure 6: Facial detection and cropped

We can then use the face detected and cropped style image in the style transfer. The crop for the facial detection bounding box is done using Numpy slicing whereas the crop and resizing for the image afterwards is done using the Tensorflow image module. As figure 9 shows, there is only a minor difference between the average image from the unchanged style image in figure 1 and the average image in figure 9. Additionally, removing the background and detecting the face together have little difference. We did manage to

adjust the color palette in the content image more accurately. However, this result is mainly visual, and is difficult to effectively quantify. We can see that the "new image" or stylized image from the CNN is still extremely distorted and without averaging out the content image and the stylized image, there would be an excess of noise in the result. The "new style image" is just an addition between the content image and style image, and is purely for measuring purposes.

To summarize the above, we found the best results came from first scaling the style image to center the cartoon face to maximize the proportion of the screen dedicated to the character style. After adding and clipping the content and style image in some linear equation to create the new style component, it can then be ran against the original content image. The kinds of images that the style transfer generated from this were definitely not close to the originals:

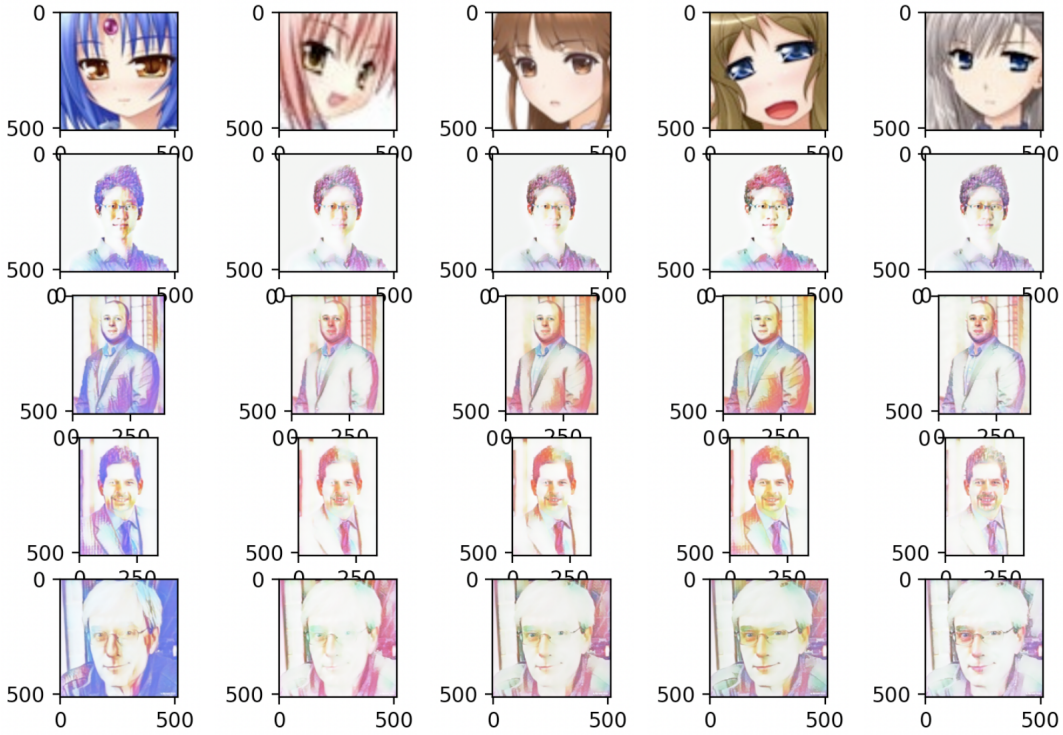

Figure 7: The resulting stylized images without averaging result with original content

However, they did retain some a component of the style that could be used. With this stylized component not in play, the average of the original and new worked to could be used to map the derived style to the original.

It is worth mentioning that in in the earlier discussion, when we found that the linear combination of components to make up the new style image has effect, we also found that varying the linear coefficients used in this equation do modify the results. For example, below is a trial run with using the same coefficients for the content and original style components:

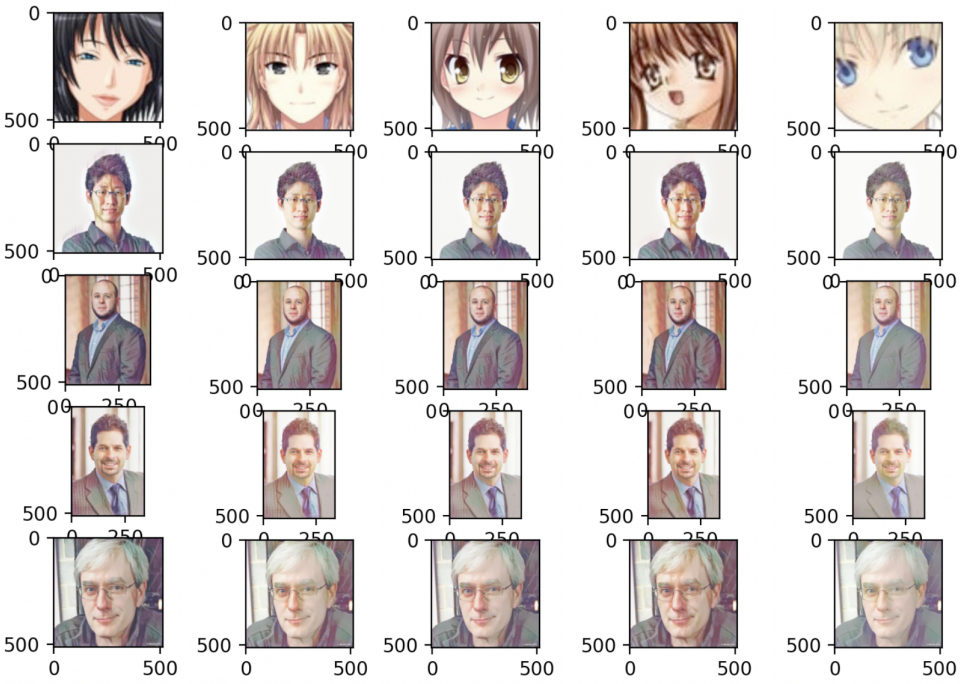

Figure 8: The resulting stylized images w/ averaging with 1 + 1 style-content image addition for new style image

However, when the style image was emphasized slightly more in the consideration (i.e. 0.9 - to - 0.7 ratio), it did have some positive effects with regards to the style:

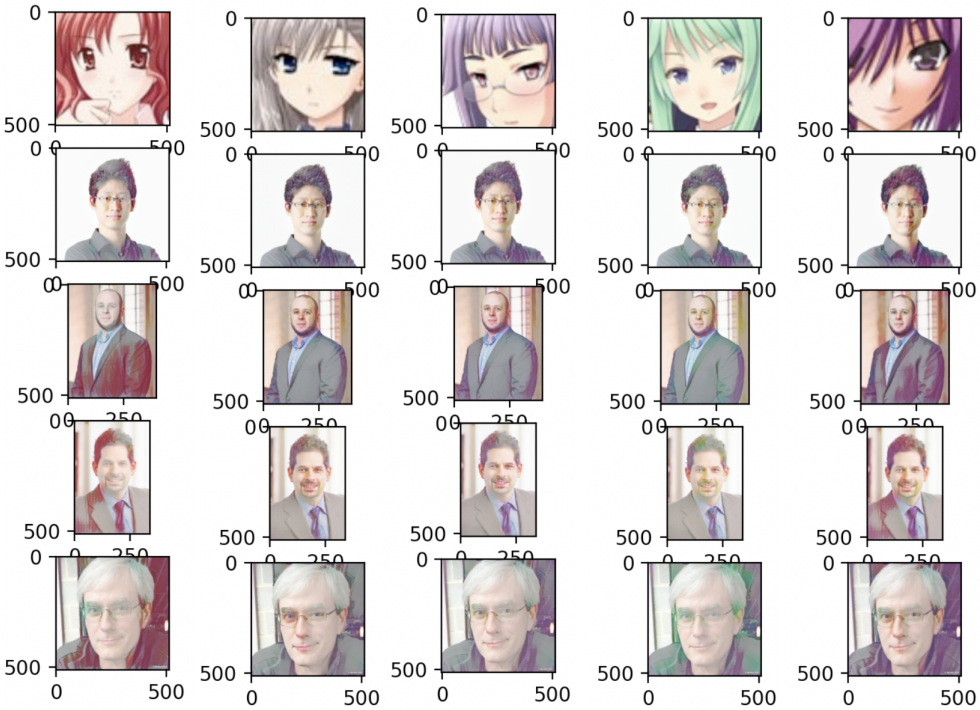

Figure 9: The resulting stylized images w/ averaging with slightly over-emphasized style image.

This of course could not exploited beyond reason, overemphasizing the content image would start to make resulting images approaching the results of pure style transfer with no modification to the new image.

6 Conclusions

Although these results were not as drastic as we would have liked, the effectiveness of altering and cleaning the input images is clearly a task that is worth further research. Finding effective datasets can be difficult and much of machine learning and artificial intelligence is dependent on having a myriad of good data. The main weaknesses of our project is the lack of substantial quantitative data to justify what is an improvement to the original style transfer. Additionally, we could have used a more robust convolutional neural network that was trained on anime images for style transfer rather than use an arbitrary style transfer model even if it was cleaner. Additionally, the GitHub repositories that we augmented to suit our task were a bit out of date so finding updated versions or implementing the functionalities of these programs using modern techniques may also improve our results. If we had more time, a possible extension of this project would be to find a better way to stitch images together while also accounting for style. In this way, fusing two images can actually be done properly.

7 Appendix

Roles:

Vadim Kudlay: Worked on arbitrary style transfer model, created skeleton program for experiments, and figure visualization

Victor Chen: Worked on image-background-removal, facial detection algorithm, and filtering datasets/experiment data compilation

References

Google Deep Dream Generator. (2020). https://deepdreamgenerator.com/.

Mckinsey666. (2019). *Anime-face-dataset.* https://github.com/Mckinsey666/Anime-Face-Dataset.

nagadomi. (2011). *lbpcascade_animeface.* https://github.com/nagadomi/lbpcascade_animeface.

Pasini, M. (2019, July). *Style transfer with gans on hd images.* https://towardsdatascience.com/style-transfer-with-gans-on-hd-images-88e8efcf3716.

susheelsk. (2018). *Portrait segmentation using tensorflow.* https://github.com/susheelsk/image-background-removal.

Wibowo, H. A. (2019, April). *Generate anime style face using dcgan and explore its latent feature representation.* https://towardsdatascience.com/generate-anime-style-face-using-dcgan-and-explore-its-latent-feature-representation-ae0e905f3974.

Yuanzheng, C., Xinzhu, M., Zhihui, W., Haojie, L., & Zhongxuan, L. (2018, August). *User-Guided Deep Anime Line Art Colorization with Conditional Adversarial Networks.* https://arxiv.org/abs/1808.03240. doi: 10.1145/3240508.3240661