WaifuBlend

XinHao Lin,
YuChao Wang,
YiAn Chen

GANs N' Roses - 2021 Paper

GANs N' Roses: Stable, Controllable, Diverse Image to Image Translation (works for videos too!)

Min Jin Chong and David Forsyth
University of Illinois at Urbana-Champaign
{mchong6, daf}@illinois.edu

Abstract

We show how to learn a map that takes a content code, derived from a face image, and a randomly chosen style code to an anime image. We derive an adversarial loss from our simple and effective definitions of style and content. This adversarial loss guarantees the map is diverse – a very wide range of anime can be produced from a single content code. Under plausible assumptions, the map is not just diverse, but also correctly represents the probability of an anime, conditioned on an input face. In contrast, current multimodal generation procedures cannot capture the complex styles that appear in anime. Extensive quantitative experiments support the idea the map is correct. Extensive qualitative results show that the method can generate a much more diverse range of styles than SOTA comparisons. Finally, we show that our formalization of content and style allows us to perform video to video translation without ever training on videos. Code can be found here https://github.com/mchong6/GANsNRoses.

1. Introduction

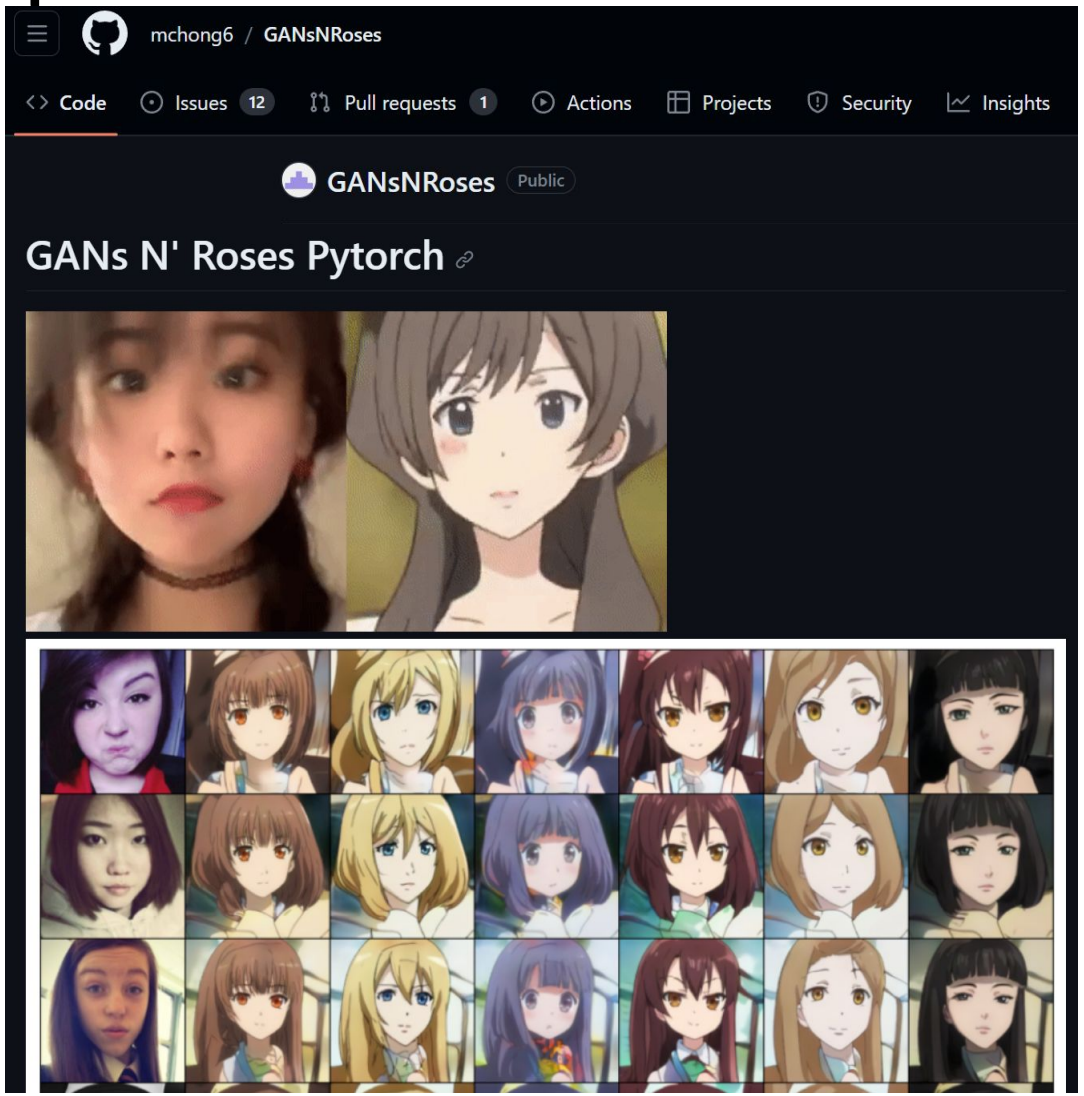
Imagine building a mapping that takes face images and produces them to anime drawings of faces. Some parts – the content – of the image may be preserved, but others – the style – must change, because the same face could be represented in many different ways in anime. This means we have a one-to-many mapping, which can be represented as a function that takes a content code (recovered from the face image) and a style code (which is a latent variable), and produces an anime face. But there are important constraints that must be observed. We want **control**: the content of the anime face can be changed by changing the input face (for example, if the person turns their head, so should the anime). We want **consistency**: different real faces rendered into anime using the same set of latent variables should clearly match in style (for example, if the person turns their head, the anime doesn't change style unless the latent variables change). Finally, we want **coverage**: every anime image should be obtainable using some combination of content and style, so that we can exploit the full range of possible anime images.

Our method – GANs N' Roses or GNR – is a multimodal I2I framework that uses a straightforward formalization of the maps using style and content (section 3.1). Achieving our goals requires carefully structured losses (section 3.3). The most important step is to be exact about what is intended by content and what is intended by style. We adopt a specific definition: content is what changes when face images are subject to a family of data augmentation transformations, and style is what does not change. This definition is very powerful. Our data augmentations involve scaling, rotating, cropping, *etc*. Thus, the definition means that content is (in essence) where parts of the face are in the image and style is how the parts of the face are rendered.

This definition allows us to learn a mapping from face images to content codes. We then pass the content codes to a decoder, which must produce anime from them and a style latent variable. It is also very important that the anime produced from a given content code "know" what that code is, and we use a decoder to recover the code from the anime. But one face should yield many anime faces, and one anime face should have the same content as many faces. This means we cannot require 1-1 loop closure of images (in contrast to CycleGAN), but must close the loop on content codes instead. This creates a difficulty, as the method might try to ignore the style code to obtain better cycle consistency on content. We show how to ensure the method produces the correct distribution of anime for a given content, resulting in a method that can produce very diverse anime. This diversity is important in applications; for example, a user might wish to exploit the control that our style code offers (Fig. 7) to get an avatar with just the right eye shape or color. Our contributions are:

- Our definition of content and style is easily operationalized; we show how to use it to ensure that the anime produced from a single content code are prop-

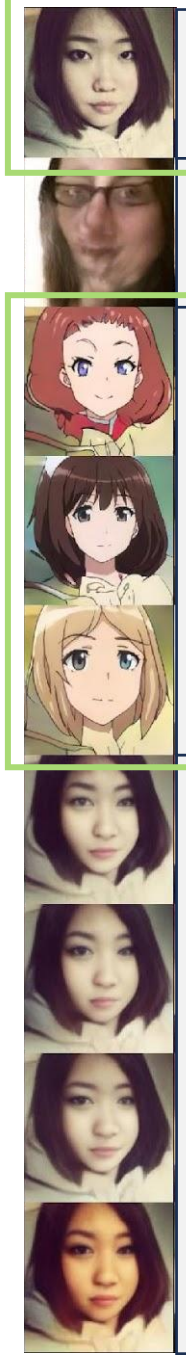
https://arxiv.org/abs/2106.06561
https://github.com/mchong6/GANsNRoses



Model Setup and Comparison

- We trained the model from the ground up with 160,000 iterations.
- Parameter differences shown below:

	GAN's N Roses	WaifuBlend
Batch Size	7	4
Style Consistency Loss	10	50 (default)



Original Image A

Using A as B, set as input to B2A

Using A as input to A2B

Using B as input,
put into B2A,
generate back to A

Focus on A2B, row 3-5!

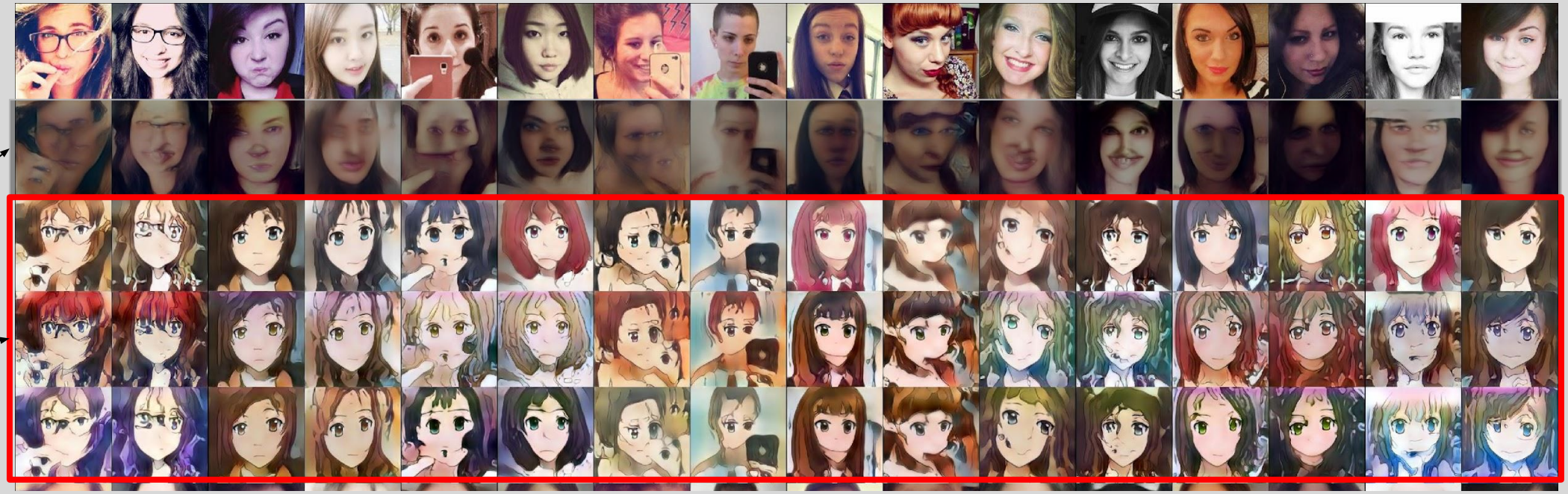
In the next few slides, we will show images about the progress as we iterate.

Our main focus will be solely **Using A as input to A2B**.

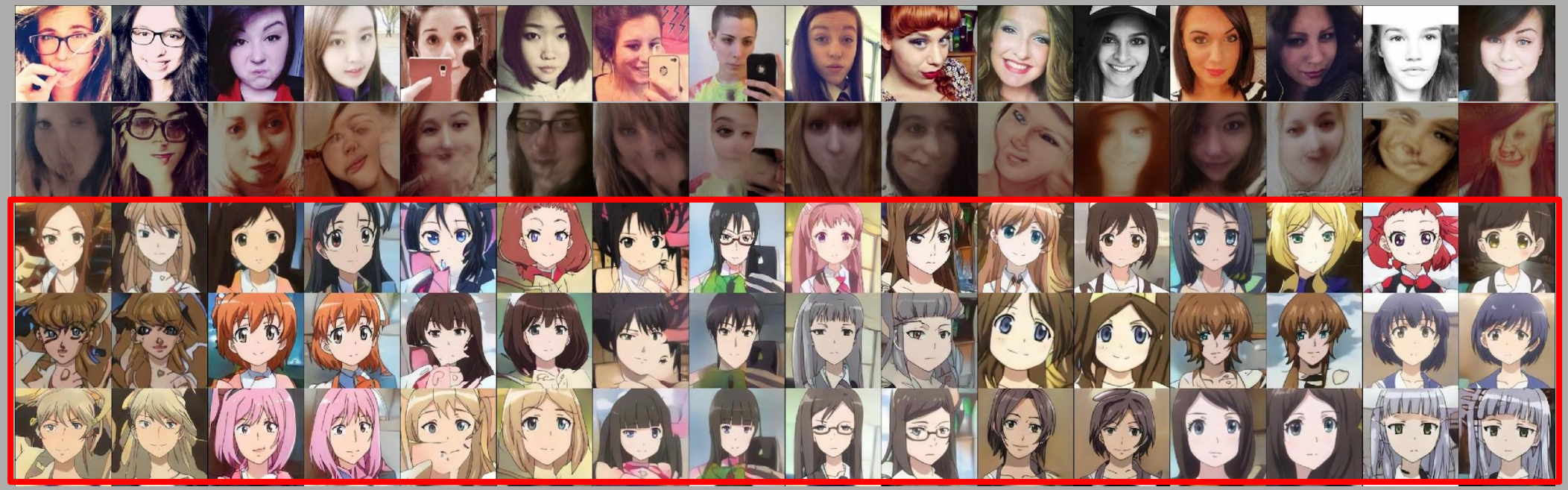
Step = 10,000

Forget about B2A

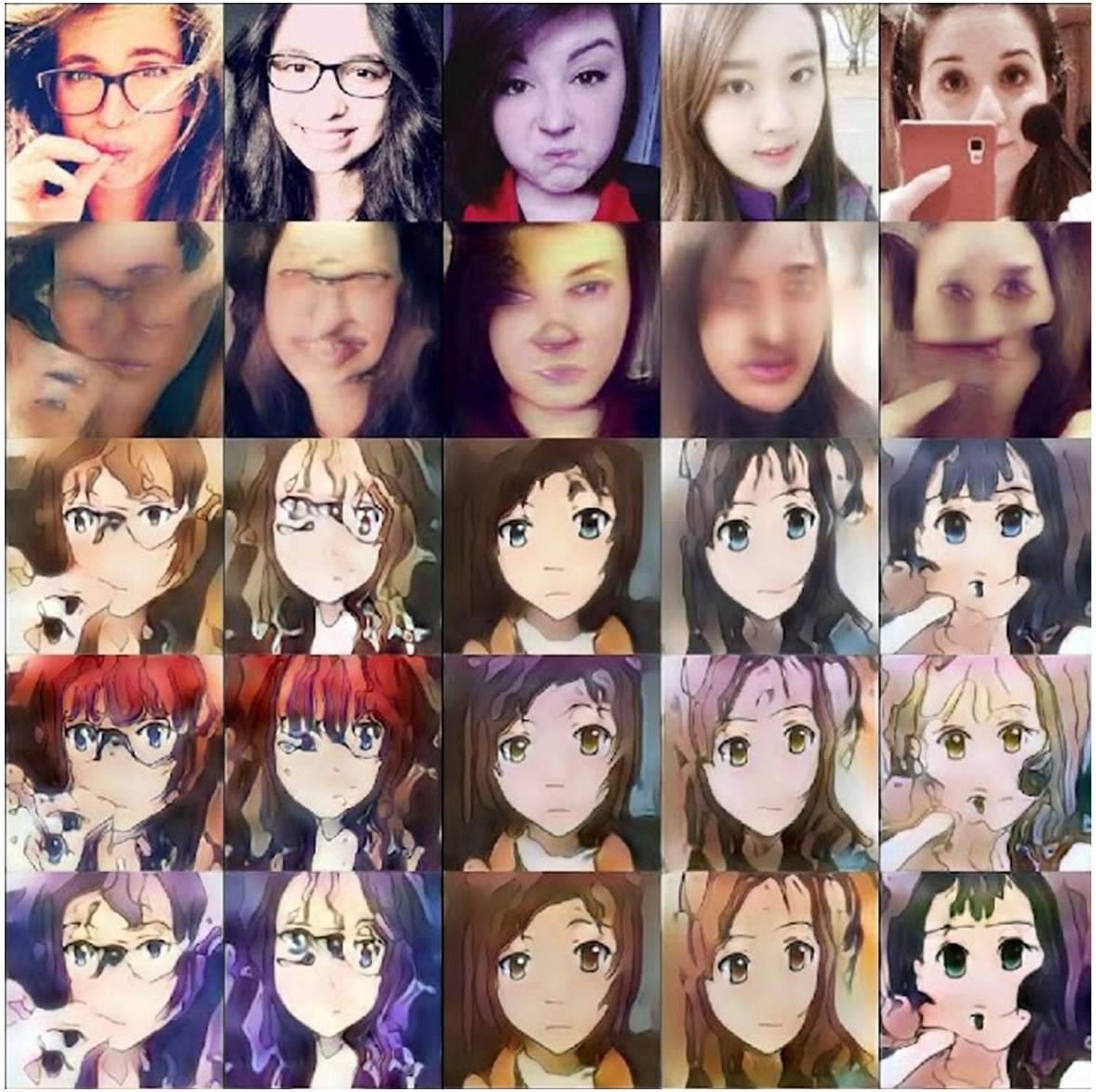
We focus on A2B, which is the part within the red box



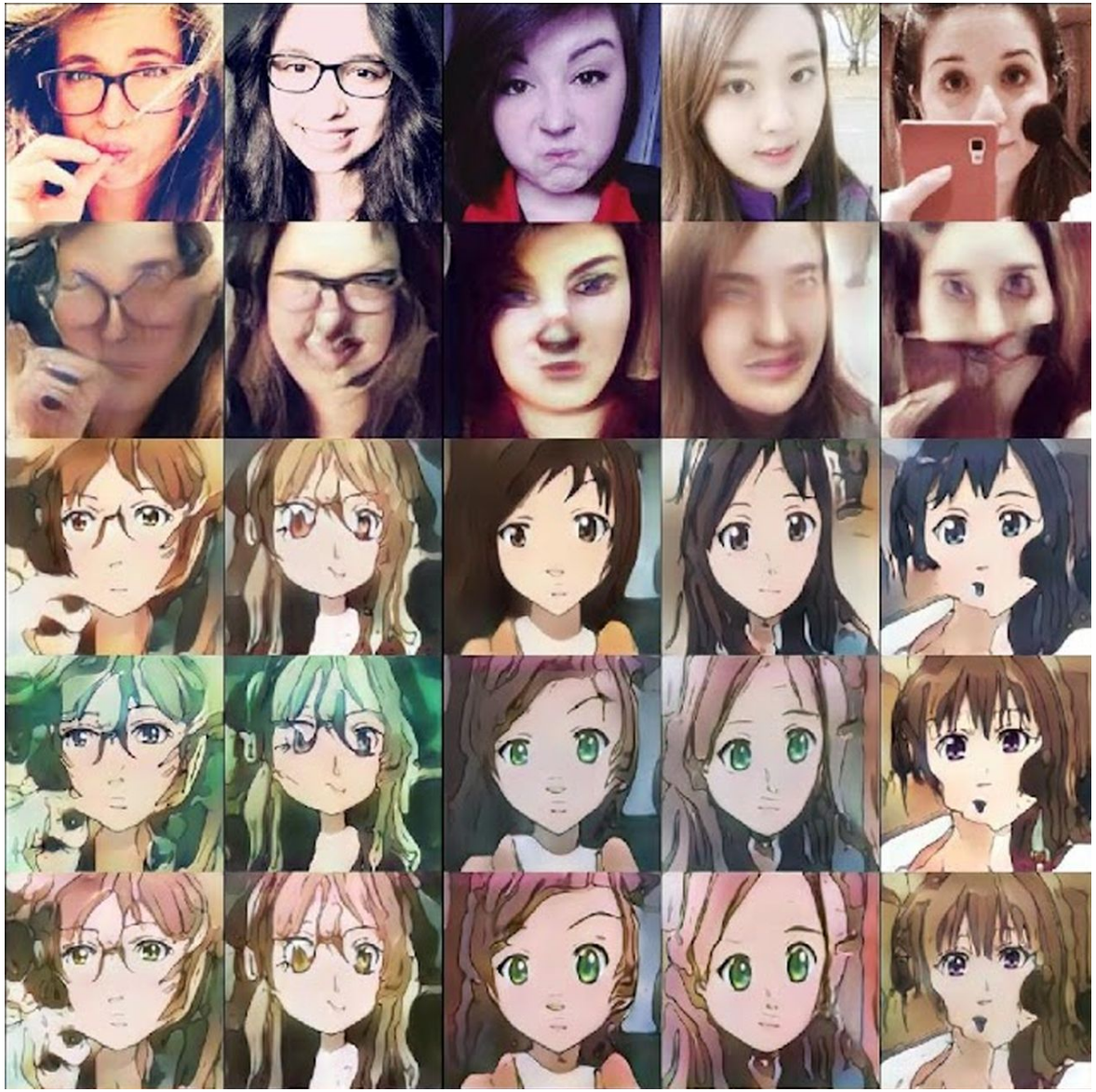
Step = 160,000



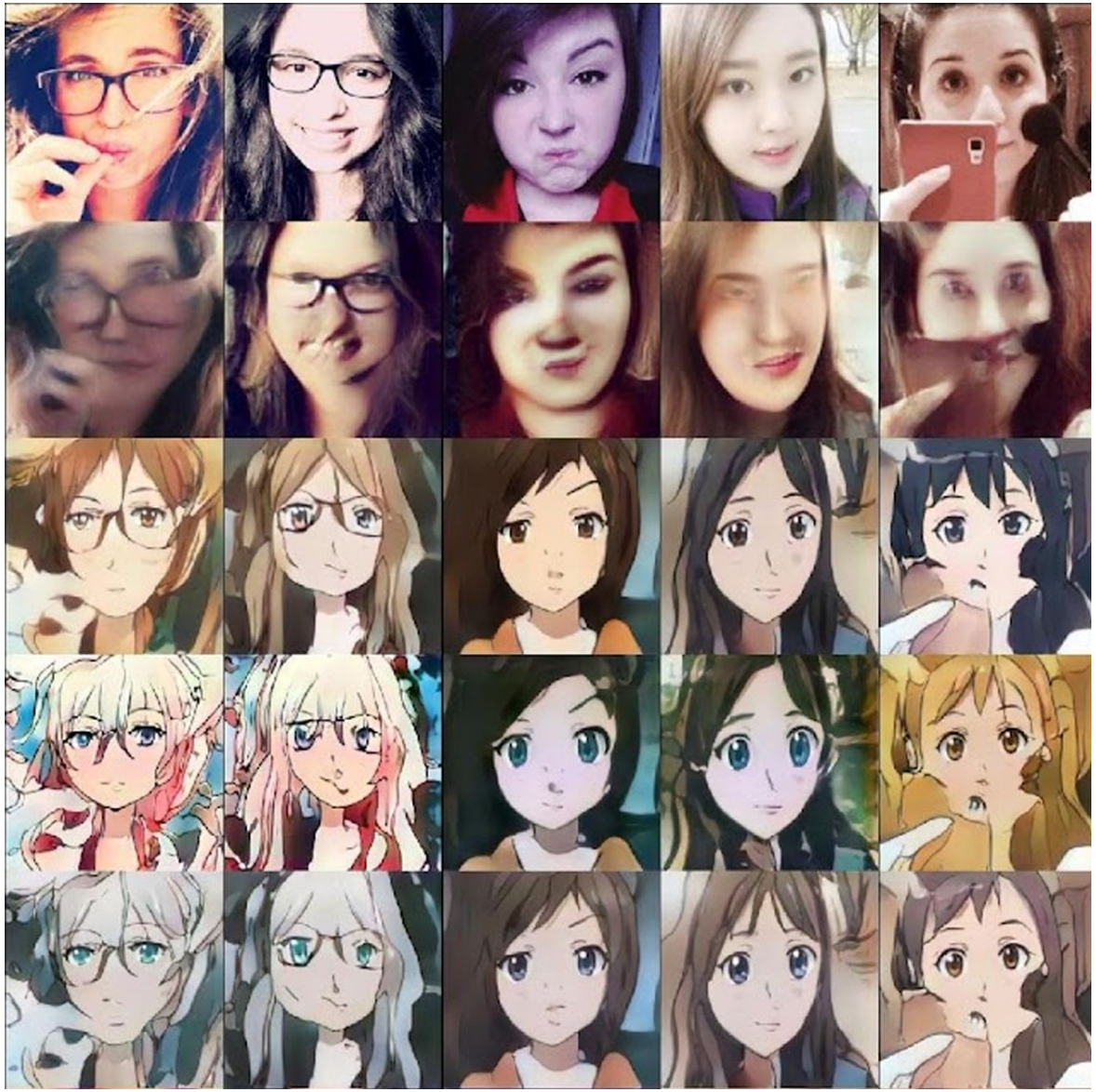
Iteration Step
= 10,000



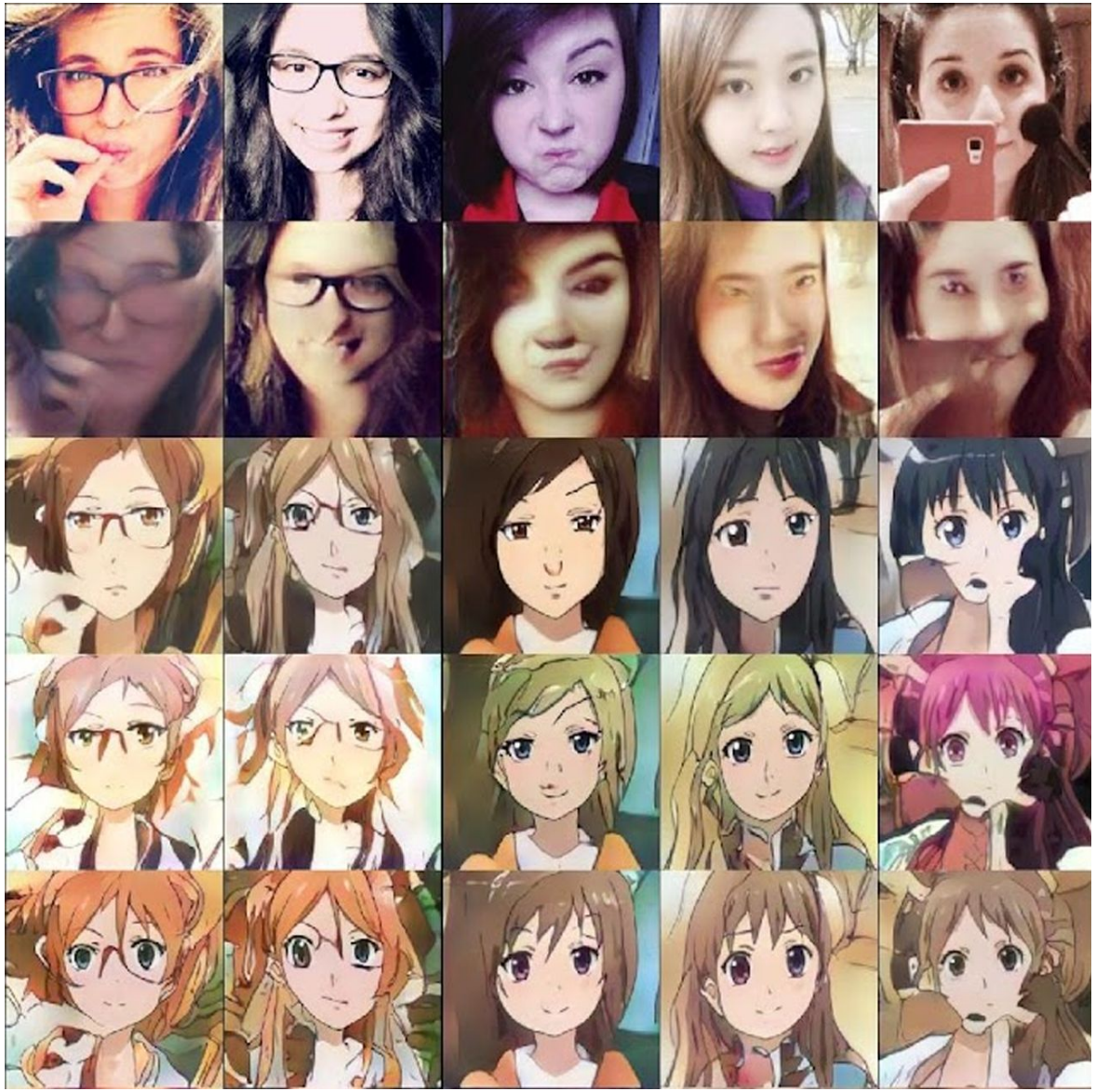
Iteration Step
= 20,000



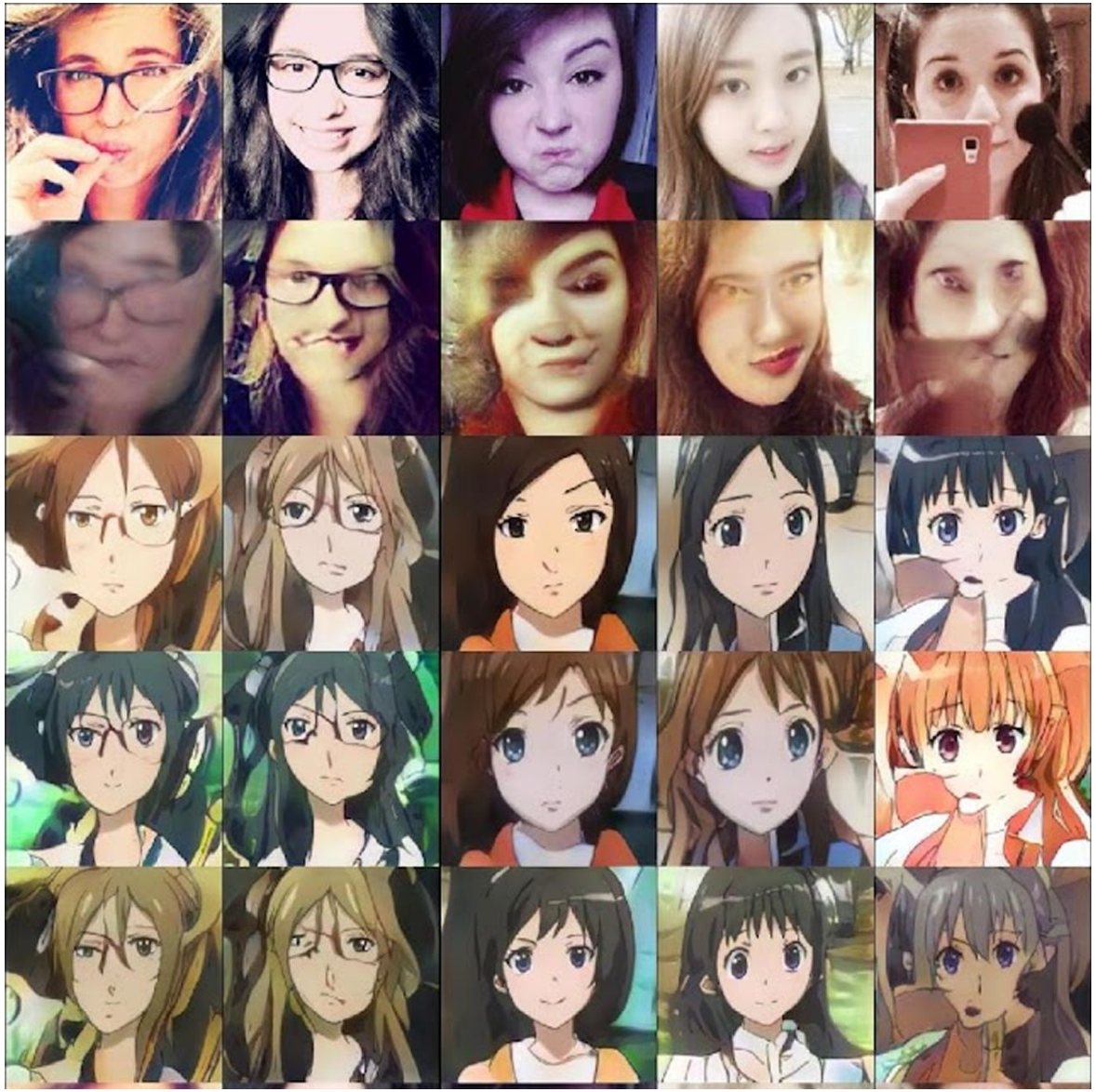
Iteration Step
= 30,000



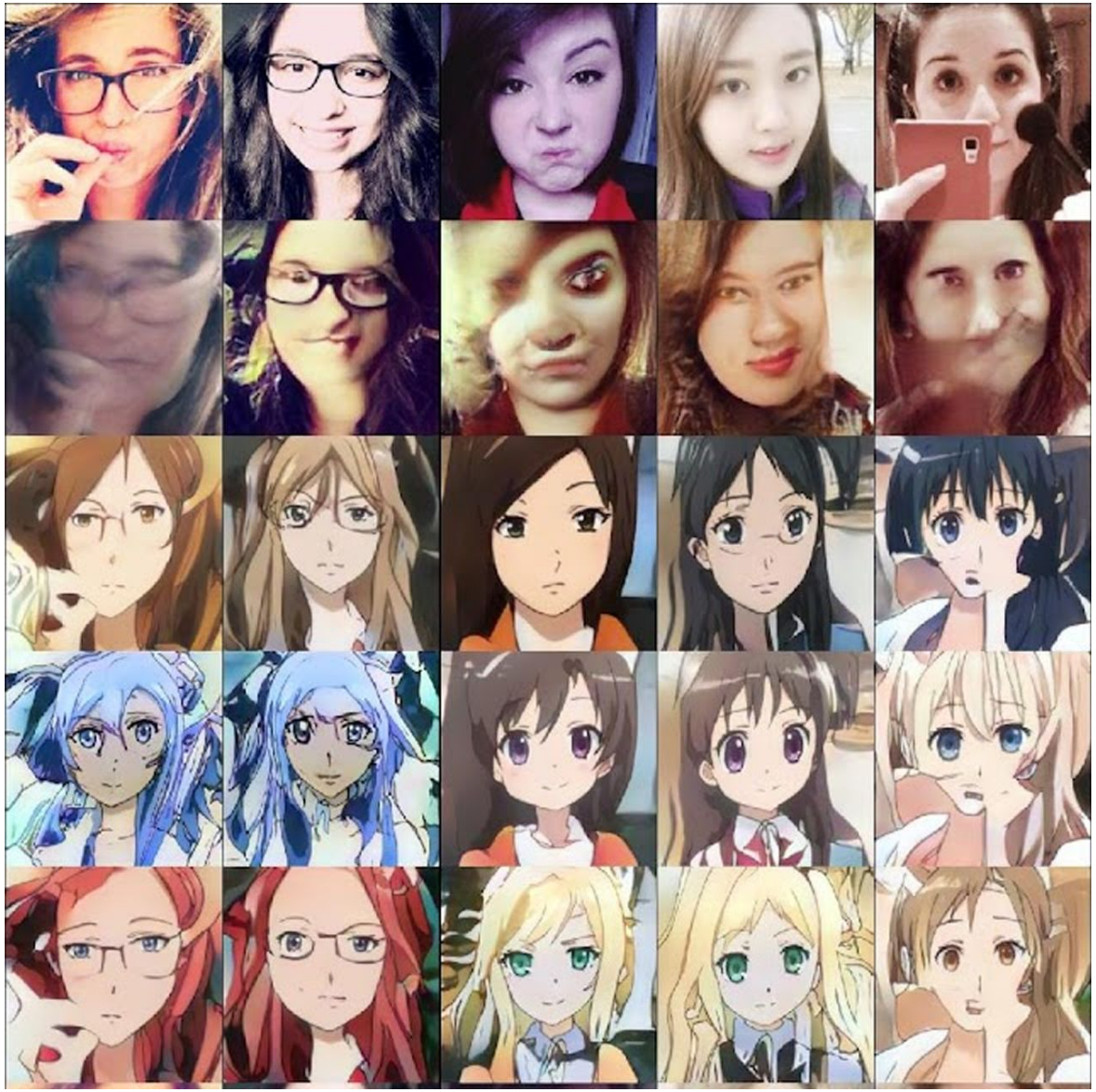
Iteration Step
= 40,000



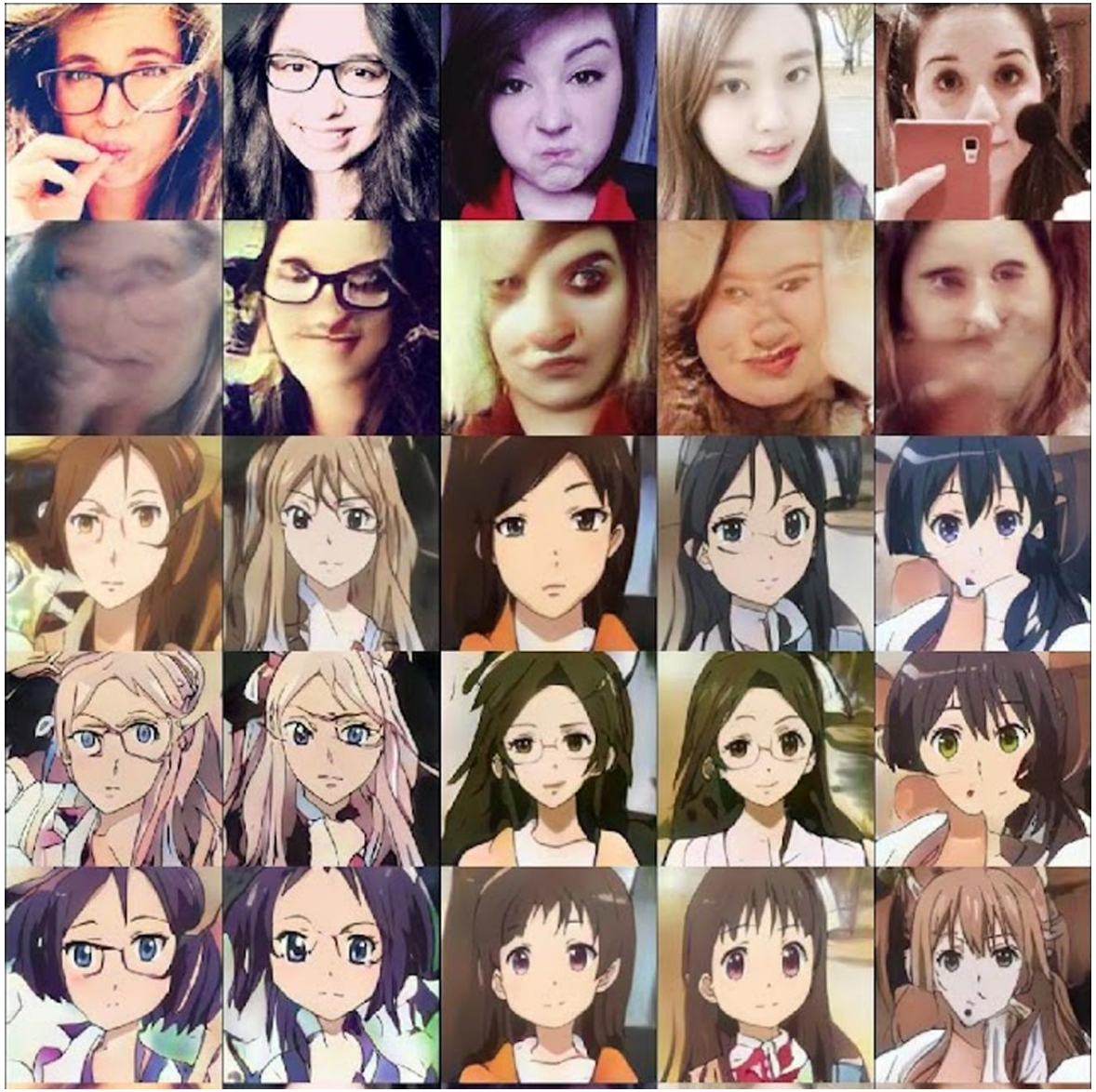
Iteration Step
= 50,000



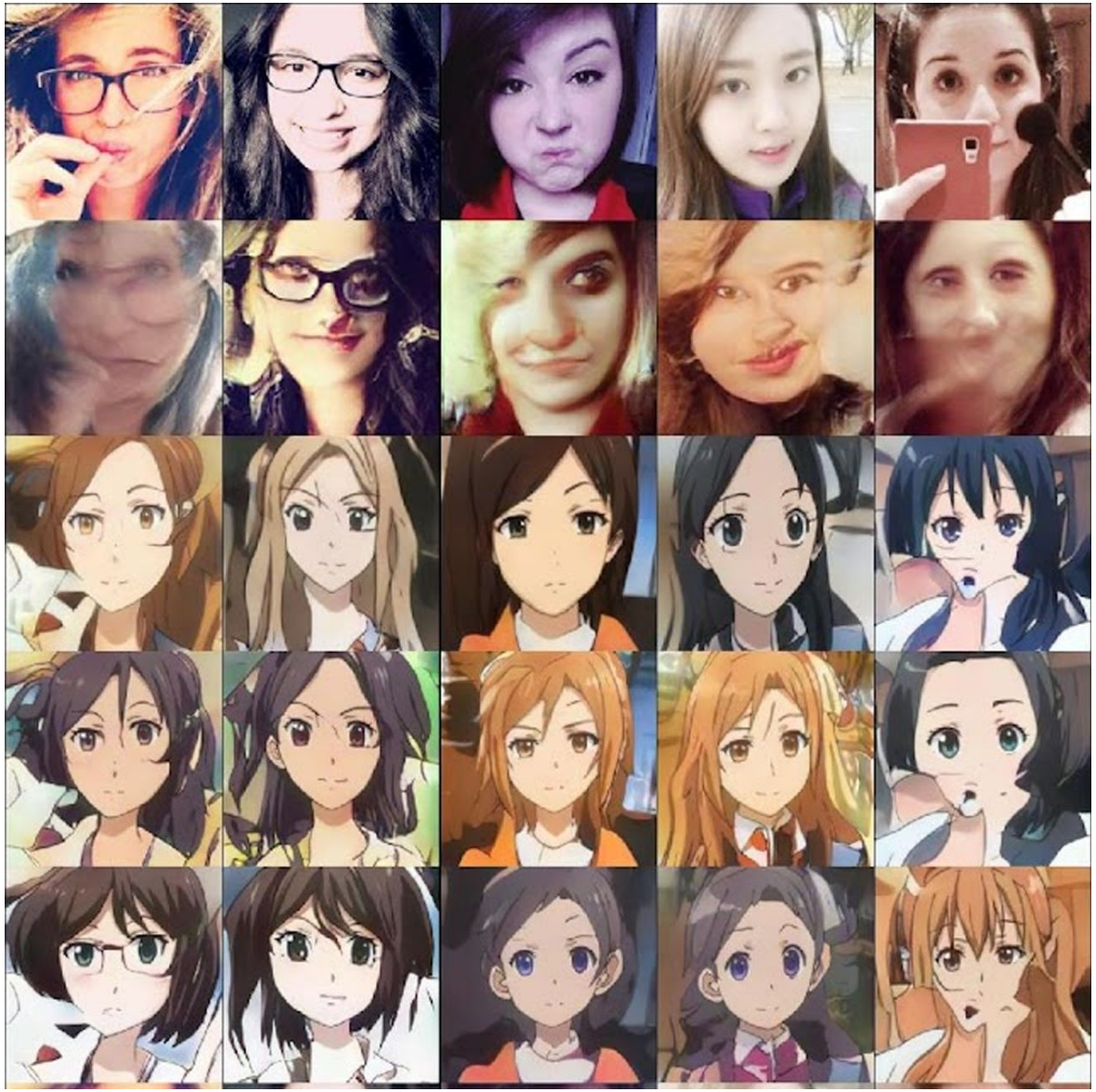
Iteration Step
= 60,000



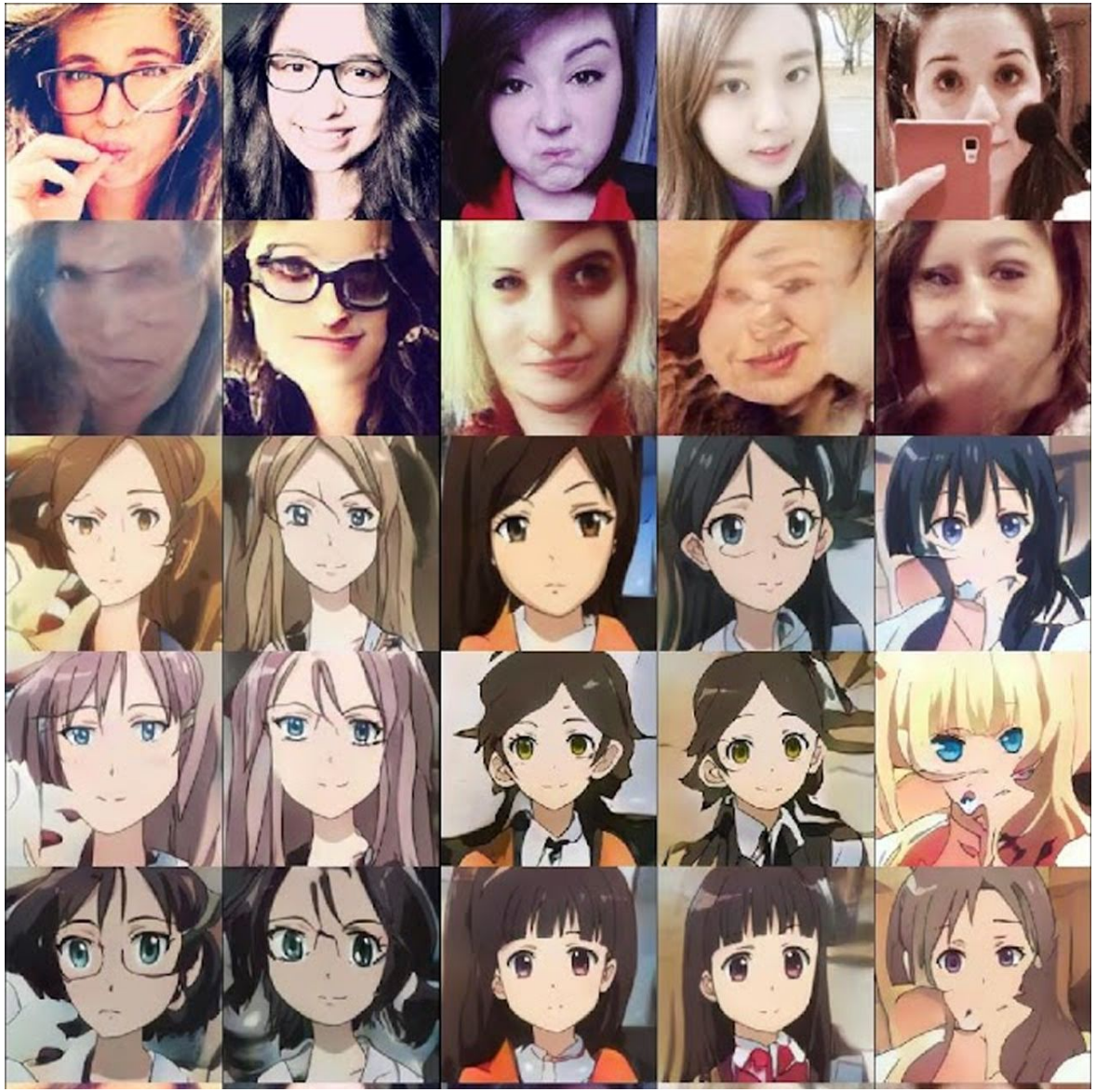
Iteration Step
= 70,000



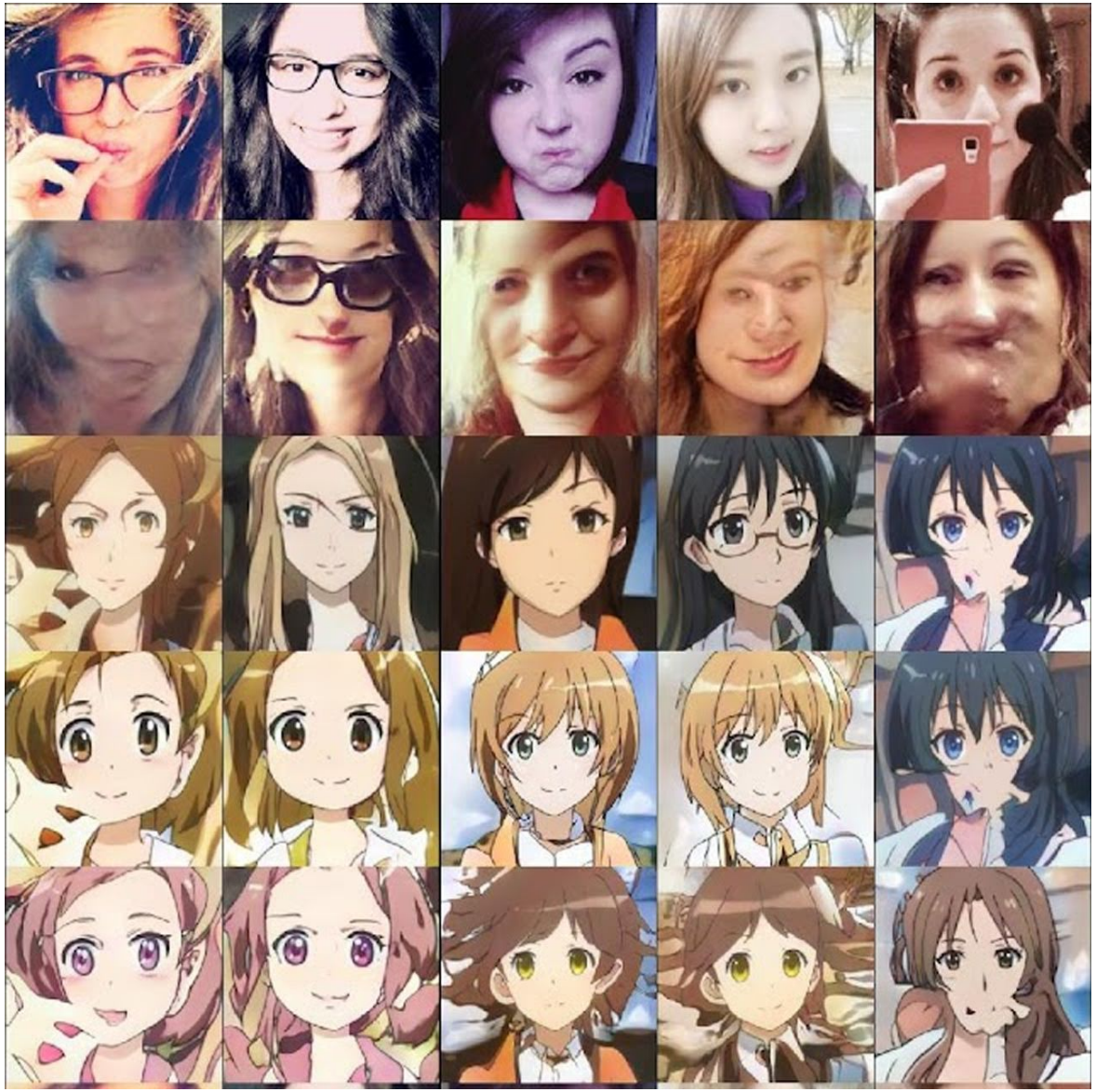
Iteration Step
= 80,000



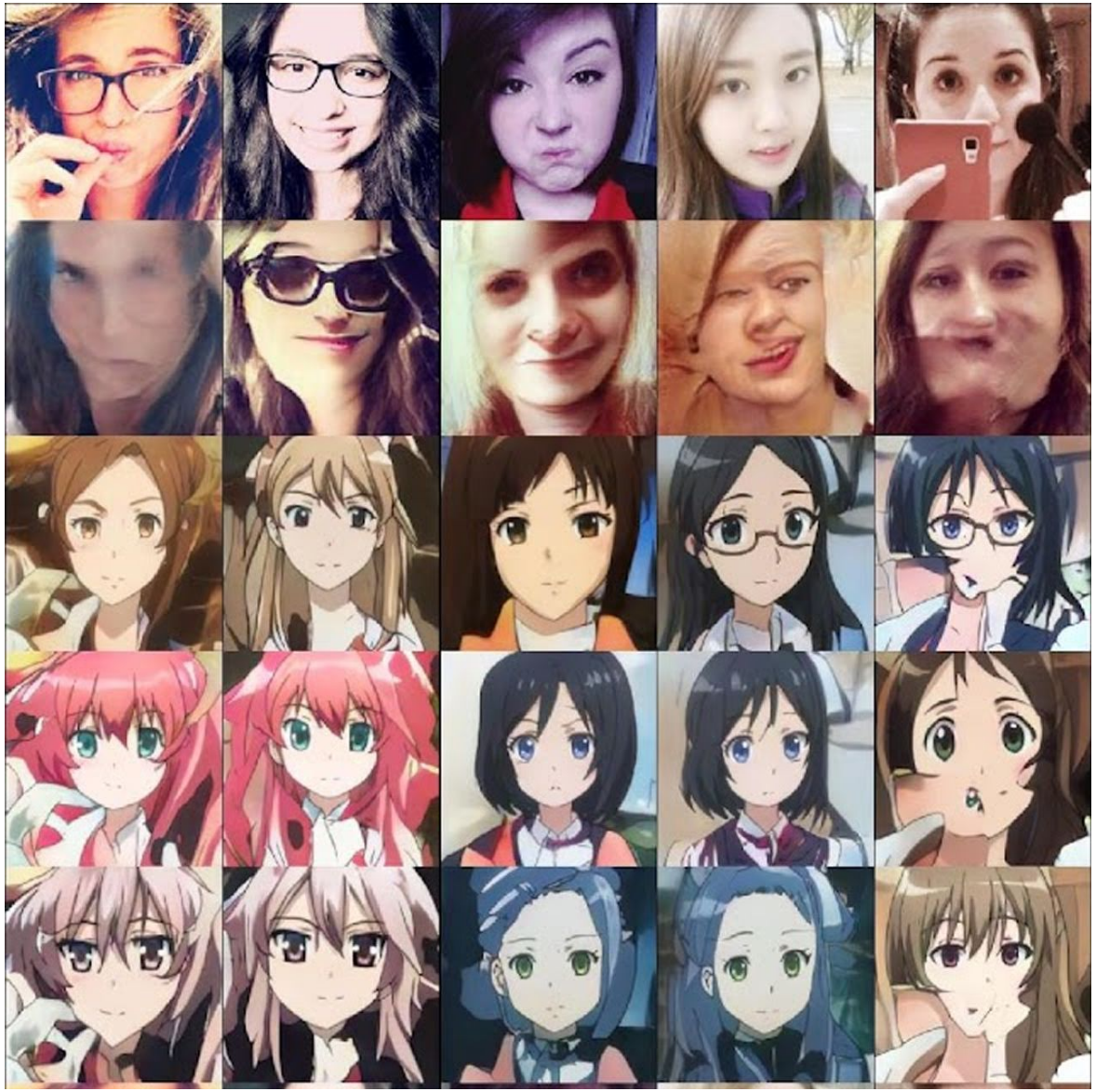
Iteration Step
= 90,000



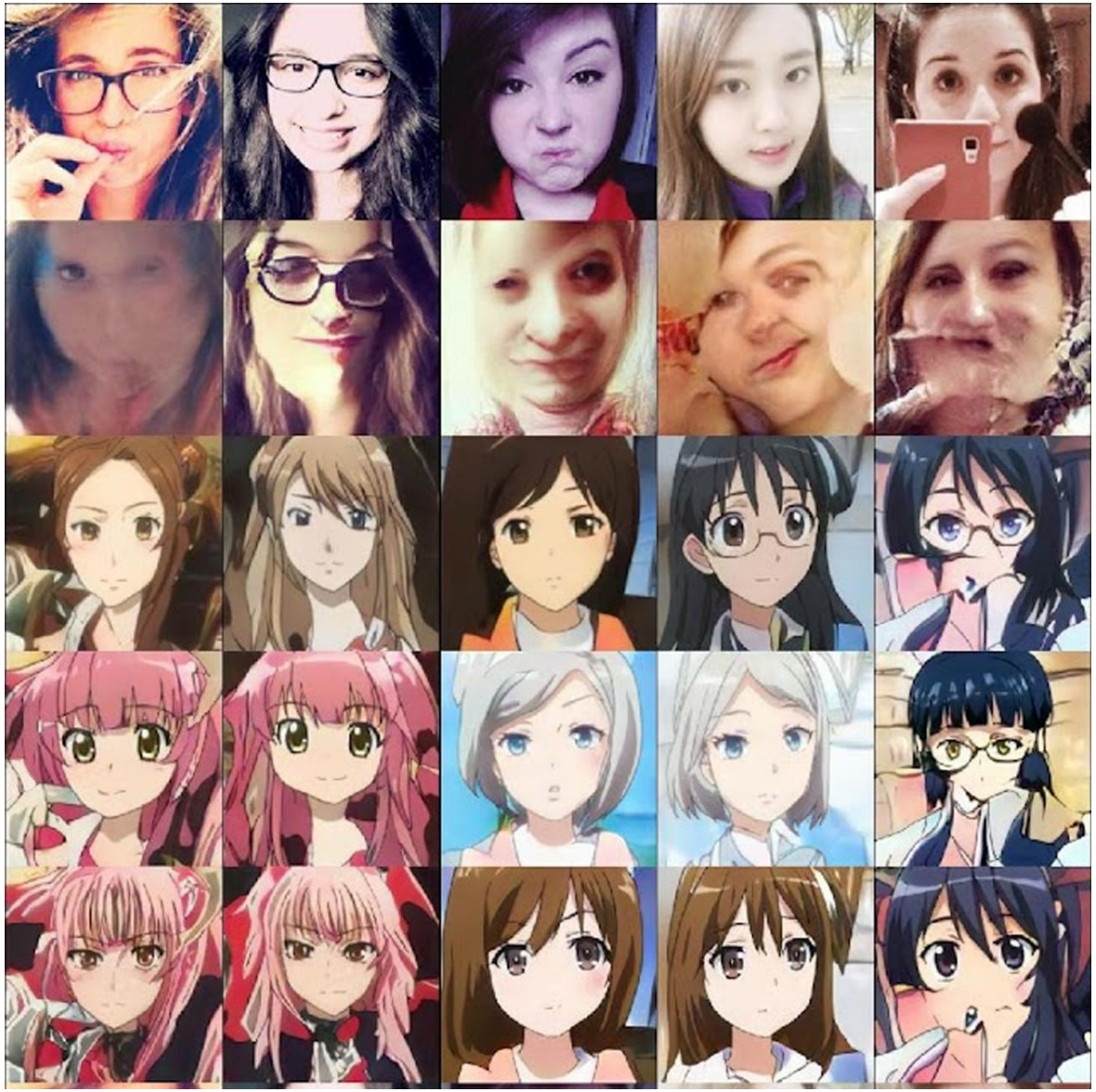
Iteration Step
= 100,000



Iteration Step
= 110,000



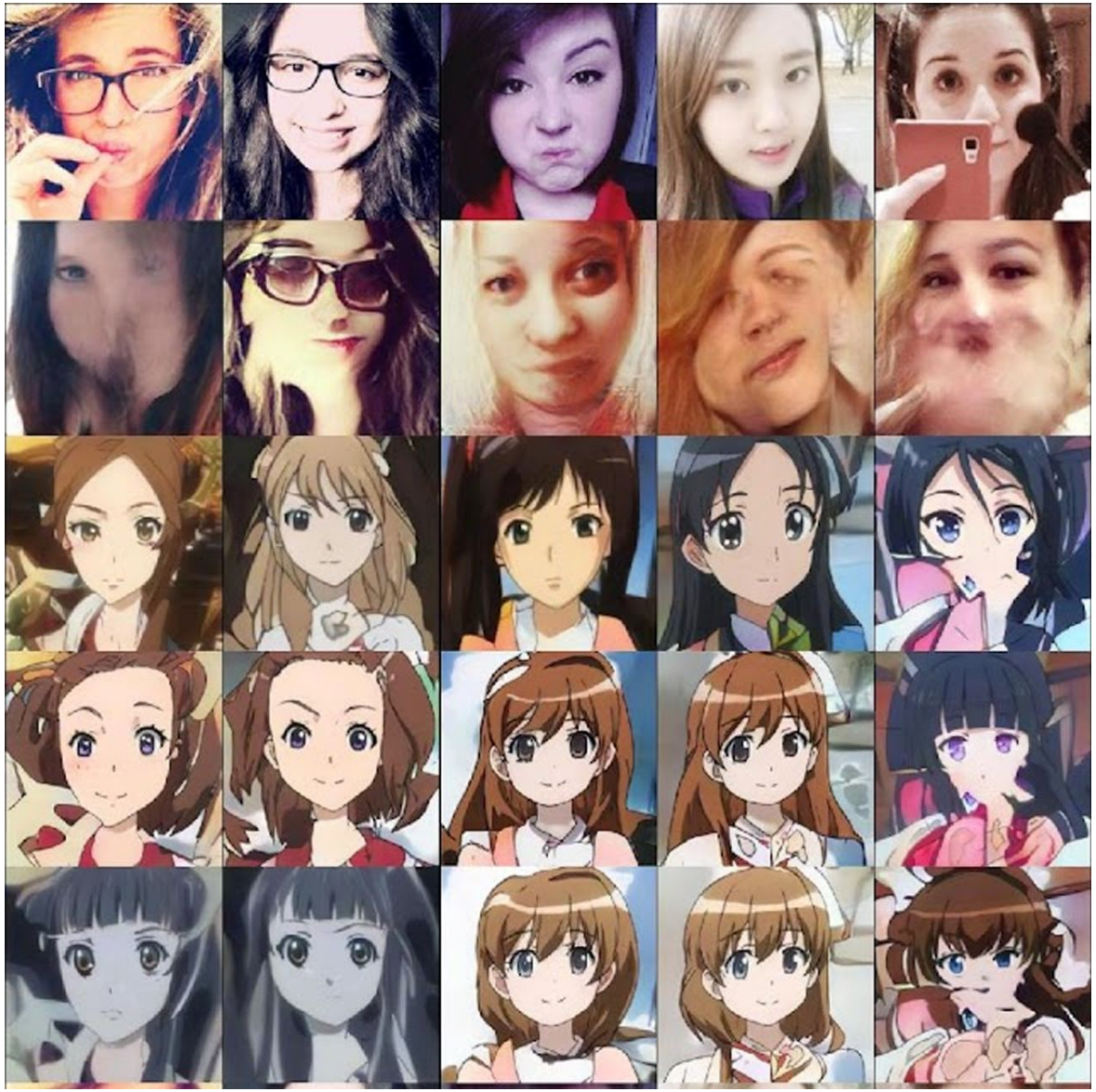
Iteration Step
= 120,000



Iteration Step
= 130,000

Iteration Step
= 140,000

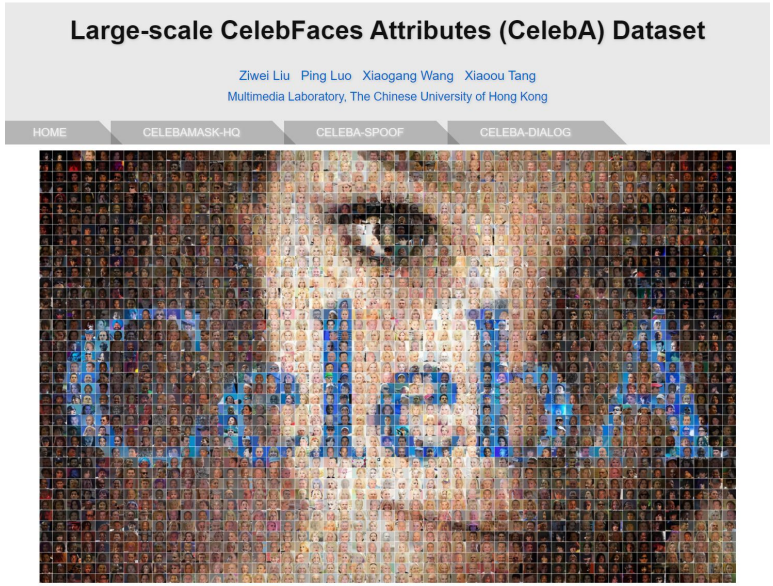
Iteration Step
= 150,000



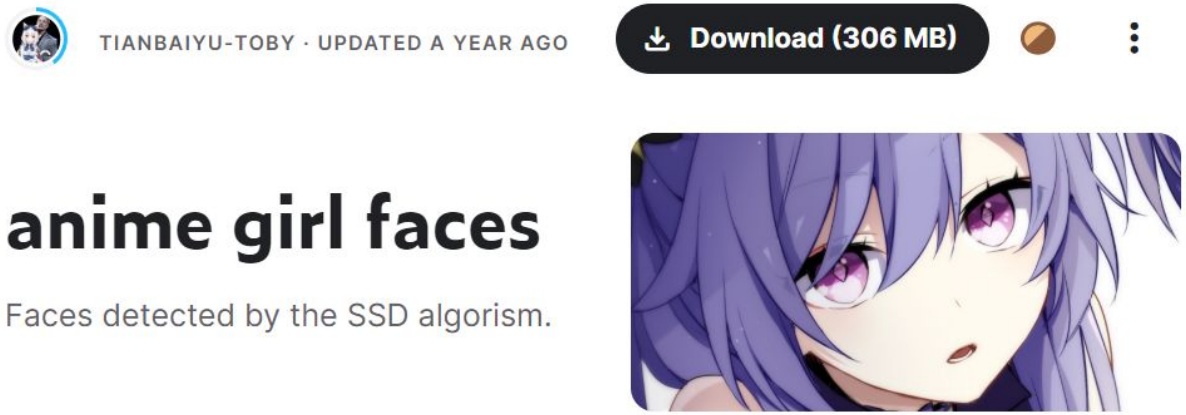
Iteration Step
= 160,000

Adding Datasets and Further Iterations

- We used other datasets to extend the generalization ability of the model.
- We fine-tuned the model with extra 150,000 iterations based on the former result.



https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html



https://www.kaggle.com/datasets/tianbaiyutoby/animegirl-faces/data

Attempt #1: Adding “anime girl faces” Dataset

The images of animated characters in the original dataset were extracted from **screenshots of anime**, predominantly facing the camera with relatively **flat shading** and **generalized details**.

In contrast, the "anime girl faces" dataset was generated through **Stable Diffusion**, resembling **detailed individual illustrations** rather than standard anime-level images. This dataset exhibits richer details, wider variety of angles, and partial facial features covered by hair.

We hypothesized that this would enhance our final output.

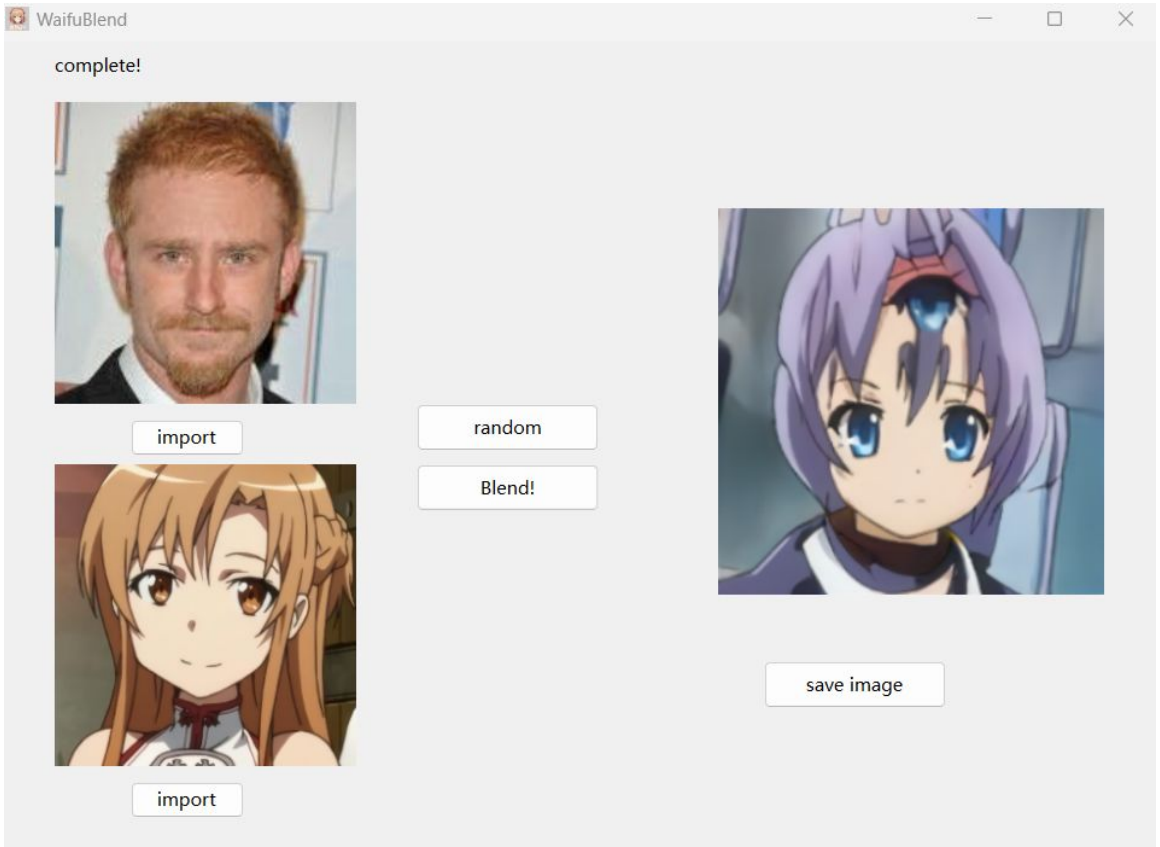
Original Dataset

New Dataset - “anime girl faces”

Attempt #2: Adding “CelebA” Dataset

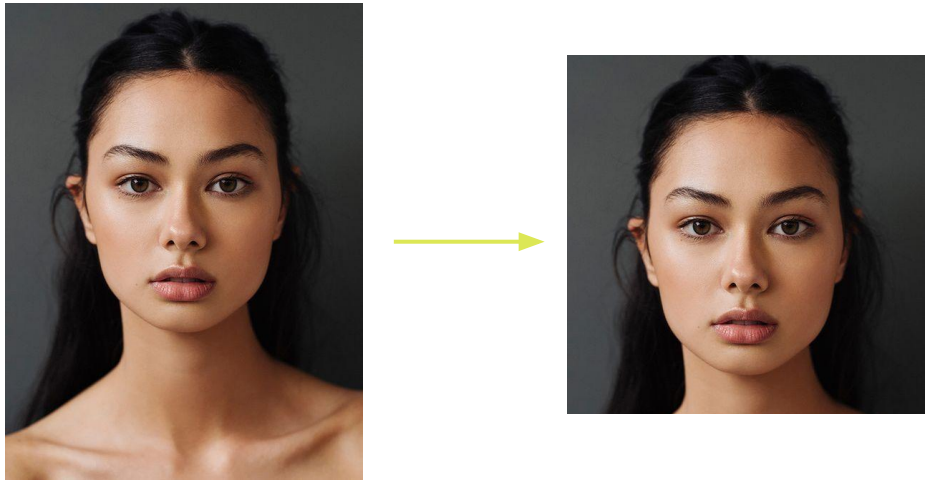
The original real-life dataset consists solely of female selfies, resulting in its near-complete inability to accurately recognize individuals with short hair or **males**. In fact, some results generated from images of real-life males has *a third eye*.

By incorporating a real-life dataset analogous to the animated character dataset, we aim to reduce the likelihood of such errors. The CelebA dataset is a suitable choice due to its extensive volume and high quality, making it apt for training.



The “Third Eye” issue with
model solely trained on original dataset

Image Preprocess



Crop images into squares for the network to work with

For images with higher resolution, we crop them to 256 x 256

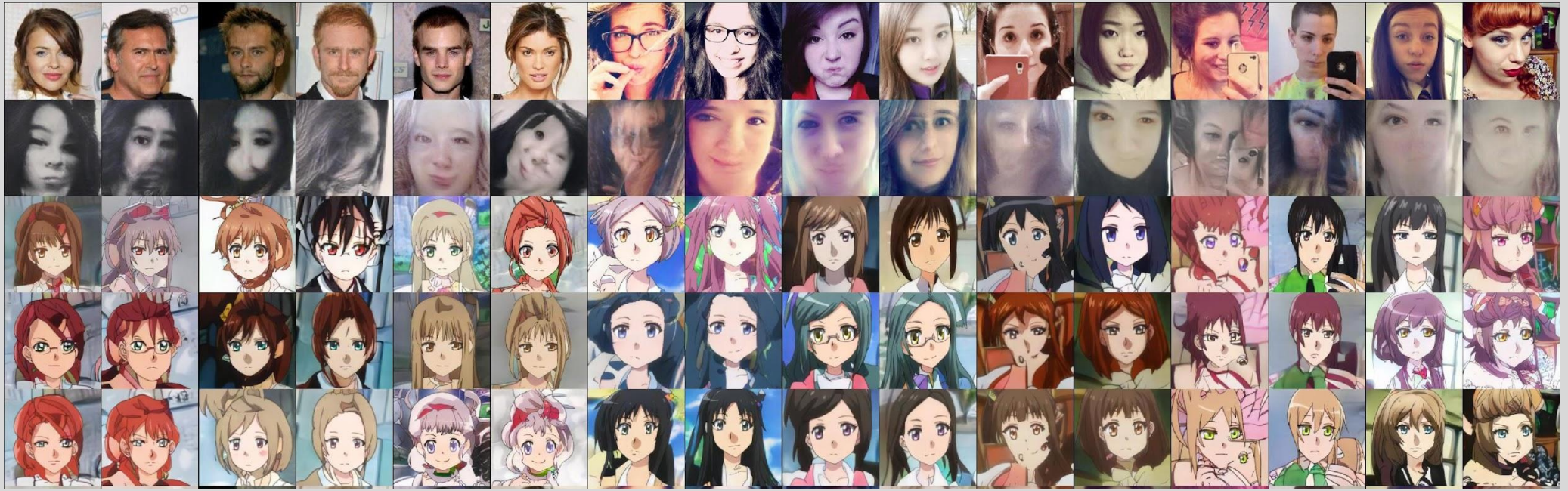
For CelebA, since the original dataset resolution is 178 x 218, we cropped them down to 178 x 178

```
test_transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.ToTensor(),
    transforms.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5), inplace=True)
])
```

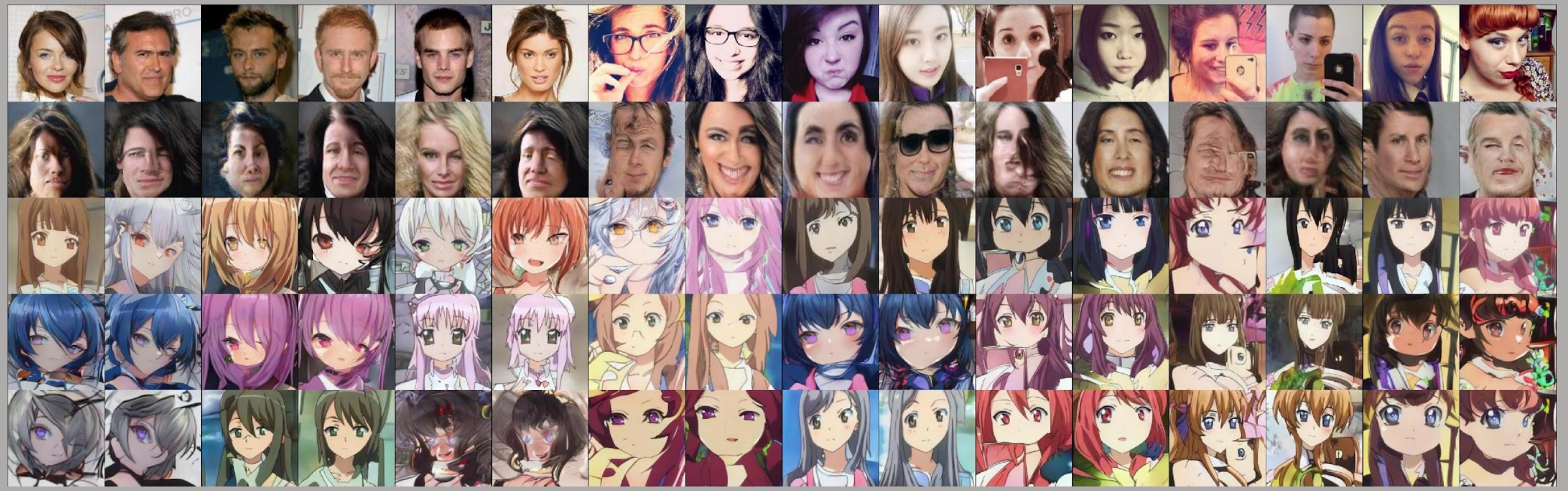
G: > Download > Test > crop.py > ...

```
from PIL import Image
import os

def crop_image(image_path, output_path):
    """裁剪图片为178x178的大小"""
    img = Image.open(image_path)
    width, height = img.size
    margin = (height - 178) // 2
    cropped_img = img.crop((0, margin, width, height - margin))
    cropped_img.save(output_path)

def main():
    directory = input("请输入图片所在的文件夹路径: ")
    for filename in os.listdir(directory):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            image_path = os.path.join(directory, filename)
            output_path = os.path.join(directory, "cropped_" + filename)
            crop_image(image_path, output_path)
            print(f"已裁剪图片: {filename}")

if __name__ == "__main__":
    main()
```

Iteration Step = 300,000



Iteration Step = 397,000



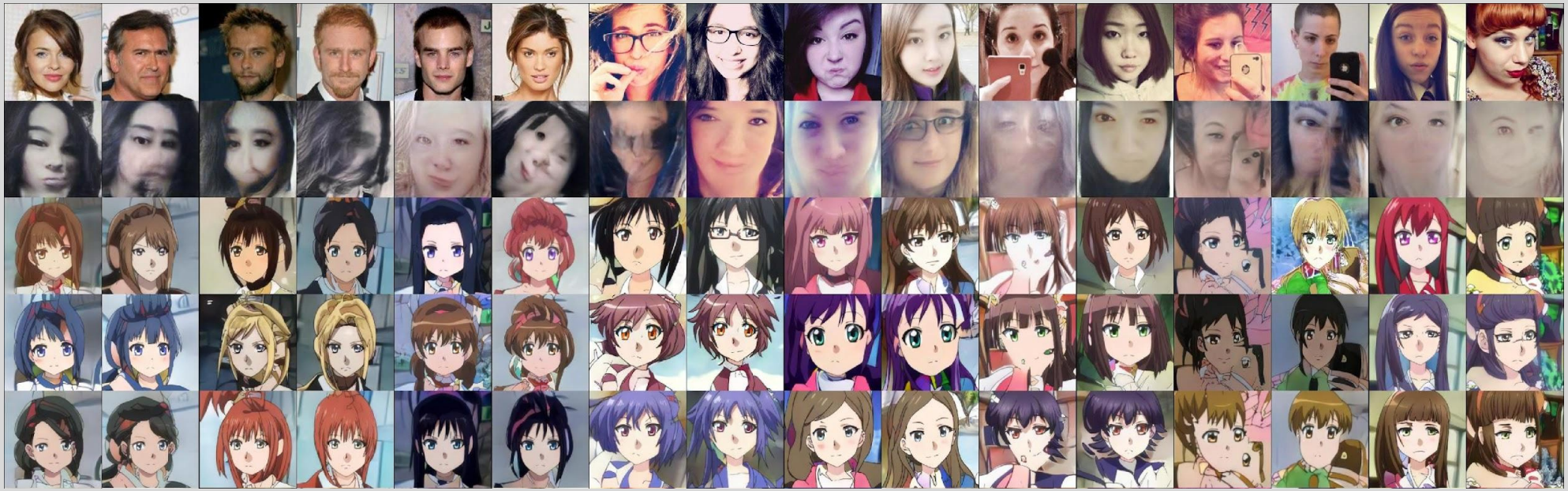
Decision to Exclude “anime girl faces” Dataset

Regrettably, the output after incorporating the "anime girl faces" dataset was inferior to the previous results. After some further training steps, the new output was markedly worse than before. The generated results typically exhibited **asymmetrical eyes** and **facial contours**.

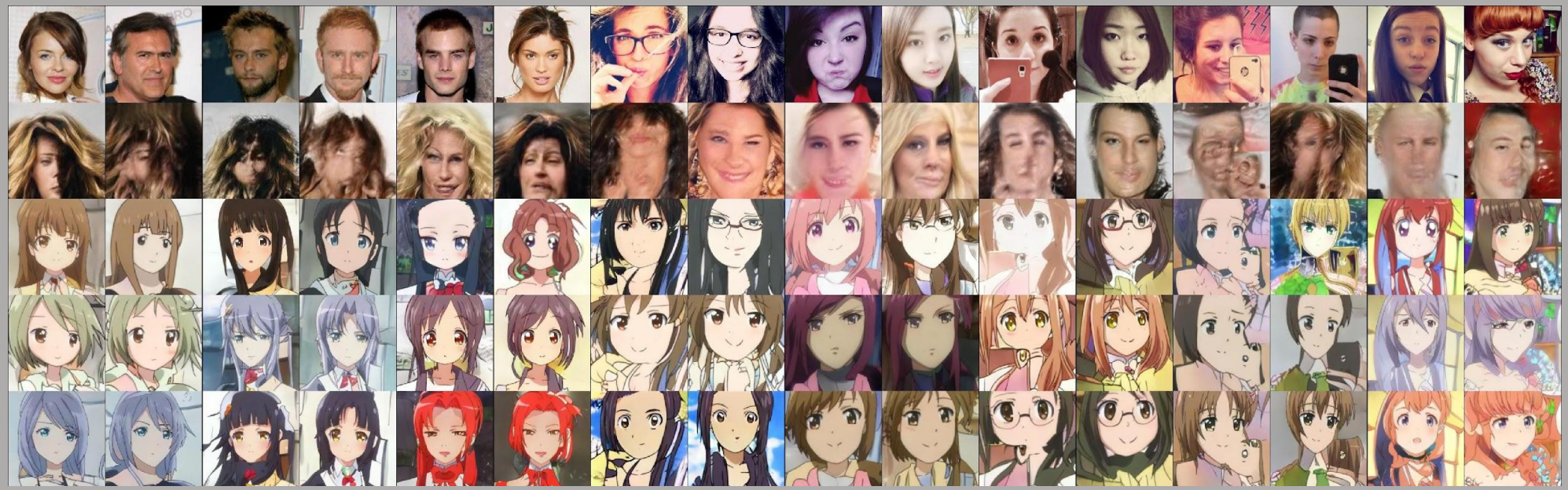
Upon further examination, we identified distinct feature discrepancies between the original "selfie2anime" dataset and the new dataset, complicating the model's ability to aptly fit and learn from the divergent dataset characteristics. The facial features in the new dataset exhibited significant variations, especially with *pronounced and exaggerated head orientations*. In contrast, both the inherent dataset of *GANs' N Roses* and the *CelebA* dataset displayed more **composed** and **consistent** facial expressions.

Ultimately, we opted to exclude the "anime girl faces" dataset and solely utilize the CelebA dataset as an addition dataset, and retrain our model.

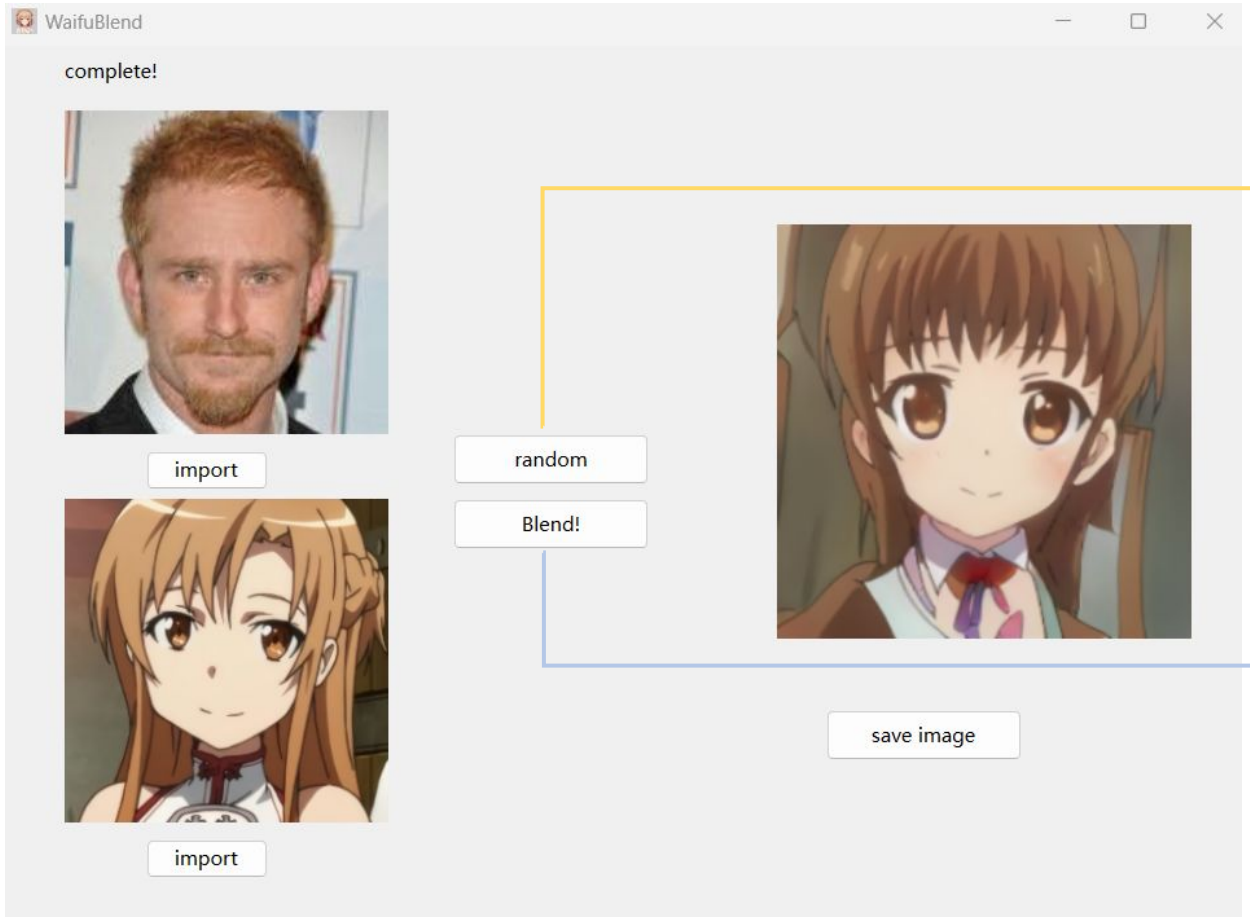
Re-Iteration
Step =
300,000



Re-Iteration
Step =
450,000



WaifuBlend Front End

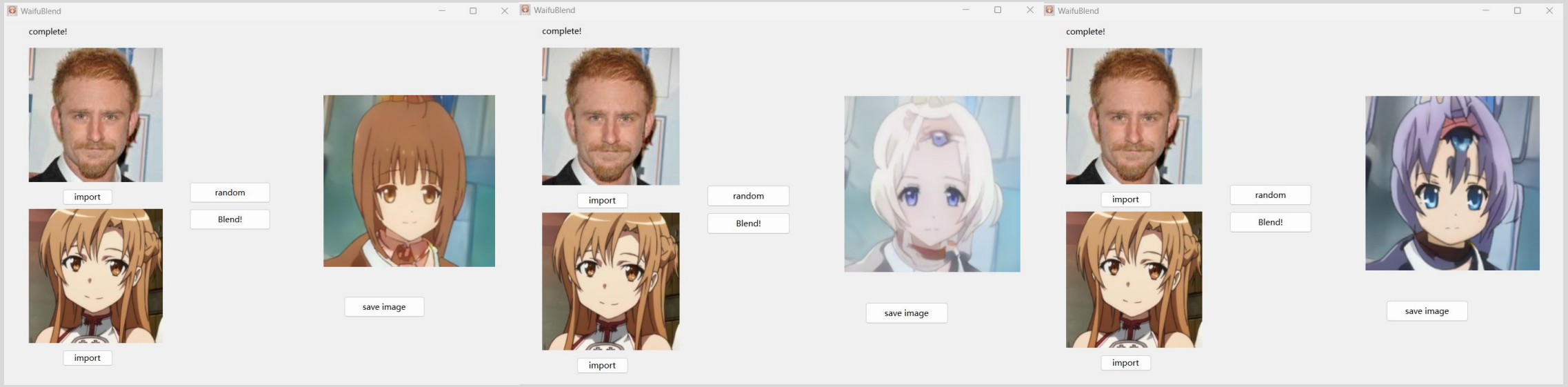


Generate anime character from random style

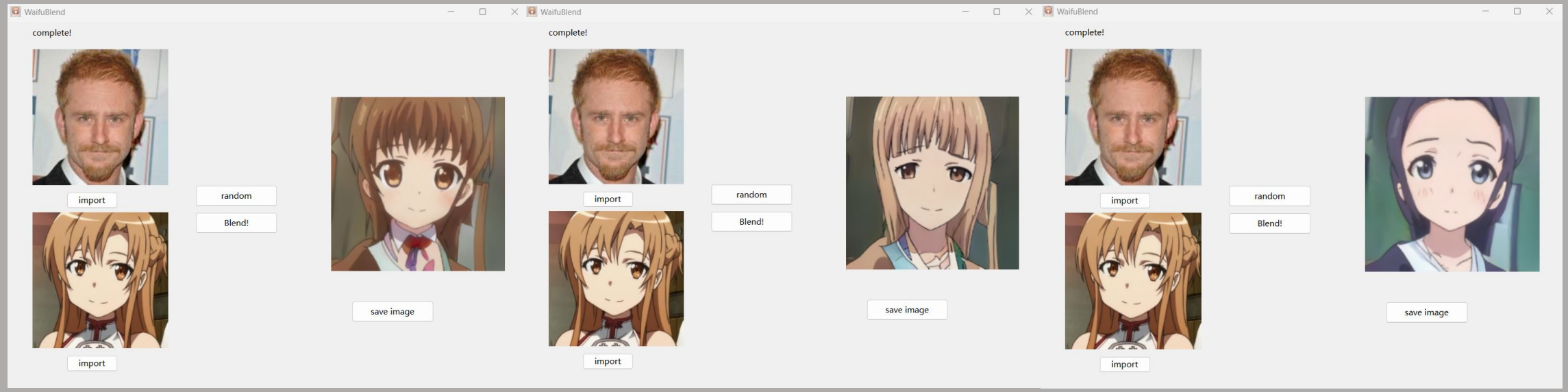
Blend anime character from given anime input

Front end created using **pyside6**

GAN's N Roses Base Model



Our Model



Conclusion

- We incorporated additional datasets to enhance the model's generalization, especially for male inputs.
- We created a user interface and implemented image preprocessing to improve both the model and user experience.

Limitation

- The model struggles to accurately recognize eyeglasses, leading to potential loss of eyeglass details or erroneous addition of eyeglasses to the generated images.
- The model also fails to correctly identify smartphones that may appear in selfies, an issue stemming from the original training dataset. The generator might mistakenly interpret the smartphone as part of the facial structure, compromising the authenticity of the generated images. Unfortunately, we have been unable to identify a valuable training dataset to rectify this issue in time.


Thank You For
Your Time!!!!!