CSC-676

Computer Vision

GAN Versus: Anime Faces

Group:

Chenxi Liao

Minh Nguyen

Sheng Kai Liao

American University 4400 Massachusetts Ave, NW

Spring 2021

Table of Contents

Abstract

In the computer vision domain, generative adversarial network (GAN) is a symbolic machine learning approach to generate artificial images. To achieve it, GAN usually contains two modules that play a minimax game, a generator that produces fake images and a discriminator that classifies the images as either real or fake.[1] Since Goodfellow (2014) set up the frame of GAN, many extended versions of GAN have been developed rapidly for various applications in different domains. In this project, we compare two GANS (StyleGANs and DCGANs), implement them on an animate dataset, and discuss their results of image synthesis.

Introduction

GAN has shown its potential in enhancing a variety of fields of research and industrial applications. However, we are particularly interested in three major usages of GAN: image synthesis, style transformation, and feature extraction.

Image synthesis is highly demanded in numerous fields. Anything from generating photorealistic images to something more specific like texture synthesis, face generation, general object generation, pos-related generation and etc.

Style transformation is a creative extension of blind image synthesis, enabling intuitive, scale-specific control of image features and high-level attributes. It can be used in many aspects such as anime character generation, age and gender prediction, and texture manipulation.

The improved quality of GAN rendered images demonstrates that the models are capable of capturing meaningful and reusable features from the supplied datasets. Although the features might not be explicitly or directly interpretable, they allow us to perform tasks such as style manipulation, data augmentation, and image classification.

As for model evaluation, we will use Fréchet Inception Distance (FID) as a quantitative evaluation method for our GAN models. The real and fake feature distributions can be approximated by two Gaussian distributions, and we can compute the distance between these distributions using Frechet distance to evaluate model quality.

In this paper, we compared StyleGAN and DCGAN, and discussed their differences and similarities from the aforementioned perspectives.

Dataset

The dataset we are using is Anime Face Dataset on Kaggle.[2] It was created by Spencer Churchill. The dataset contains 63,632 animate faces without labels. The typical size of an image is between 90×90 and 120×120. We also reduced the dataset to 19,000 images in some of our models.

Experimentation

I. DCGAN

1.1 Image Synthesis

We tested our DCGAN model performance on two scales of the dataset. We trained the model based on 1) 50 epochs of the full dataset and 2) 100 epochs of the shrinked dataset (with 19000 images). Other parameters are hold the same: learning rate is 0.0002, batch size is 128, image size is 64×64×3, size of input noise is 100, adaptive learning rate $\beta_1$is 0.5. We see that the quality of generated images is better when we use the full dataset, suggesting that the abundance of data contributed to the models capturing image features more accurately. Meanwhile, the increasing number of epochs cannot compensate for the lack of training image, resulting in lower quality of result (**see Fig.1 in Appendix II**).

By training with the shrinked dataset, the models did a fair job capturing the major attributes. In contrast, using the full dataset, the model not only learns those attributes, but also represents them more accurately and generates less artifacts (**see Table.1 in Appendix I**).

1.2. Feature Exploration

As the author of DCGAN pointed out, the manipulation of the Z representation, ie. the input noise, yields smooth transformation of certain characteristics in the image.[3] We explored this based on the model trained on the full dataset with 50 epochs. Here, we used a fixed input Z, which consists of 100 variables drawn from a standard Gaussian distribution, and manipulated the selected variables one at a time while holding the others constant, in order to observe the impact of such variables on the generated image.

We selected 10 input variables (ie. 1st, 10th, 20th…, 90th in the Z representation), and for each of them, we adjusted it by multiplying it with a manipulation factor K that ranges from -3 to 3. For example, **Fig.2 in Appendix II** illustrates the manipulation of the Z variable, and uses it as the input to generate an image. For this random input, scaling the 1st variable changes the face contour of the anime character, making the width of the face shrink as K increases. Scaling the 20th variable seems to impact the face orientation. Some Z variables appear to control multiple features, producing an effect of varying the facial expression. The 90th variable seems to impact features such as face orientation, eye position, and mouth shape, allowing the transition of facial expression and the overall appearance of the character.

However, we cannot conclude that there is an explicit mapping between the manipulation of a specific Z variable and the impact on image attributes. For example, the manipulation of the 1st variable does not always change the face contour. With another random input (**see Fig.3. In Appendix II**), the 1st variable does not have a strong influence on the face contour, and it seems to have a tiny effect on the hair style. Instead, the 60th variable now appears to have a significant effect on face shape. At the same time, the 70th variable now impacts the eye color, and we did not observe this effect in the previous example.

The difference in the effect of the variable at the same index position in the Z representation suggests that DCGAN is incapable of allowing specific-control of certain image attributes. However, DCGAN does reveal a potential of learning meaningful attributes and generating images with smoothly transiting features. Our results show the limitations of DCGAN in that the model cannot explicitly separate the features, adjusting a specific variable in Z representation sometimes lead to changes in multiple attributes and result in a global effect on the synthesized image.

1.3. Quality Evaluation

Given the limitations of computational power, we attempted to calculate FID in the training process with a smaller dataset consisting of 9000 images, and used InceptionV3 as the feature extractor. In this toy dataset, we trained on 5 epochs to get an idea of the training quality. We noticed that the FID score is always high and fluctuating. This is explainable because InceptionV3 was trained on a different dataset and might not capture the images features for the anime faces. We did not have enough time to implement a better suited CNN model for feature extraction, so we decided to comment on the results with visual inspection.[4] (see **Fig. 4 in Appendix II for plot**)

We observed that the loss of the generator and the discriminator display trends of declining in the training. However, both of them have significant fluctuations, and the loss of the generator is always greater than that of the discriminator. This implies that the quality of fake images is unstable and that the generator fails to perfectly simulate the real images. As a result, artifacts and lack of realness accompany this high generator loss. (**see Fig. 5 in Appendix for plot**)

II. StyleGAN

2.1 Image Synthesis

Our StyleGAN implementation involves selecting the first 19,000 images from our full dataset of 63,632 anime faces. We cloned NVIDIA StyleGAN GitHub and used some of the scripts as starter codes while editing only the critical lines.[5] Our images were also resized, converted to Tensorflow records (tfrecords is required since StyleGAN uses TensorFlow) and pre-processed before training our model for 3,500 iterations. After 9 hours of training, we were able to produce a model with a FID of 39.4008 which is quite decent considering our limited hardware and time. (**see Fig. 6 in Appendix II for training progression**)

As you can see the model images look quite pleasant, considering our model was trained on Google Colab Pro (1 GPU) for less than one day using only 19,000 images. The original NVIDIA's StyleGAN paper used two main datasets for two different models: Flickr-

Faces-HQ (FFHQ) and CelebA HQ, 70,000 1024x1024 and 200,000 with mixed resolution images, respectively.[6] Additionally, our model used a resolution of 64x64 which was much lower than that of NVIDIA's (1024x1024).

2.2 Feature Exploration

The beauty of StyleGAN is its controllability. The traditional GAN (Goodfellow et al.,2014), operated like a blackbox where random noises go in and an image gets generated. In this project, we explored two controlling methods of StyleGAN: **Style Mixing** and **Truncation**.

Style Mixing allows us to embed styles at different levels of our generative layers to control various features. Using this method, we were able to transfer style from raw images to our destination images. This method works quite well even on our small model with a shallow network of only a few layers. (**see Fig. 7 in Appendix II for visual explanation)**

At the coarse layer (4x4), we noticed changes in face shape, pose, hairstyle and mouth. These are the bigger, more noticeable features.

At the middle layer (8x8), we noticed changes in eye brows, eye color, nose and hair texture. These features are starting to head towards the subtle direction.

At the fine layer (64x64), we noticed slight changes in color scheme, sharper hairline and shading. These features are definitely more subtle compared to our coarse layer but not too far from our middle layer due to our shallow network.

Truncation tricks involved in controlling the W with Ψ, using this equation:

$$W_{new} = W_{avg} + \Psi(W - W_{avg})$$

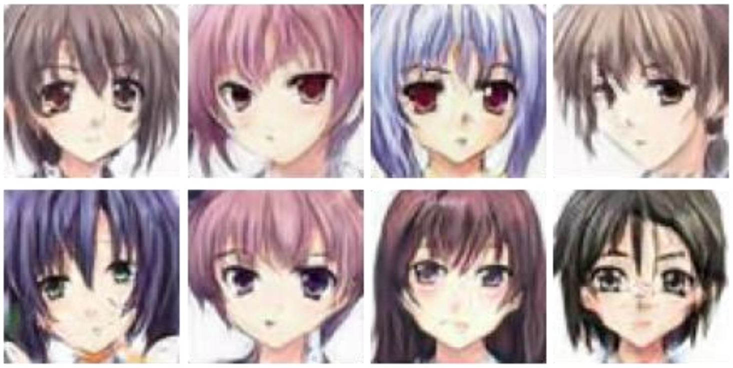
By truncating W we were able to drastically change our generated images. At Ψ =0, faces converged to the "mean" face and looked the same as our generated images without truncation. However, when we apply negative scaling to styles, we got the corresponding opposite or "anti-face." Lastly, at higher values of Ψ, gender, hair length, coloring are flipped. (**see Fig. 8 in Appendix II for visual explanation**)

2.3 Quality Evaluation

StyleGAN performed quite well on a limited dataset with limited GPU and training time. We believe training the full dataset for multiple GPU days and with all layers (up to 1024x1024) will lead to much better results in both FID and resolution. The paper used FID as the main measurement of quality so we only measured the FID value for our model which was 39.4008.

Overall, our shallow model produced results inline with the StyleGAN paper. This indicates that StyleGAN is applicable to non-natural images like animated and cartoon characters. We find the truncation trick to be especially interesting since a slight tweak in Ψ can lead to very drastic changes. StyleGAN is quite powerful, even on our simple model. Below are 8 newly generated images from our StyleGAN mode:

I.



Conclusion

By comparison, DCGAN is trained faster but it generated less satisfactory results compared to StyleGAN. DCGAN still delivers unreasonable output with relatively lower quality even when we trained the model with the whole dataset. For feature learning, DCGAN seems to extract coarse attributes such as hair color/style, eye shape and face contour and some of the features are entangled. In contrast, StyleGAN can divide attributes into three levels, coarse, middle and fine. In this case, the features can be defined in certain levels and we can manipulate the input variables at various levels in order to generate images with specific requirements of details.

Overall, both models have their own advantages and drawbacks. For DCGAN, training is relatively easier but the model is inflexible for training datasets with higher image resolutions and is incapable of controlling the style of our generated images. For StyleGAN, the training process requires a lot of computational power and in some cases, the output images may contain inconsistent style mixing leading to random artifacts since different style variables were embedded to the intermediate layers. However, due to the different needs in various aspects, we believe that both models can find their suitable applications.

Individual Contribution

Chenxi: I helped with setting up the training of DCGAN on Colab, inspecting the features extracted from the model (see Section 1.2). It is interesting to explore how and to what extent meaningful features are learned from the network. I also helped a bit with the implementation of style mixing in StyleGAN.

Minh: I did extensive research on StyleGAN and worked mostly on StyleGAN implementation including pre-processing the dataset, editing the original StyleGAN's scripts, training our model on Colab and generating new images. With the help of both Chenxi and Sheng Kai, we were also able to generate the videos and style mixing figures. I also took care of some of the final editing and formation of our deliverables.

Sheng Kai: Initially, I was responsible for implementing the dataset into StarGAN. Unfortunately, since StarGAN required attribute labels for each image which is hard to achieve in a short time, we decided to drop off StarGAN. For this project, I contributed to solving the technical issues from Chenxi and Minh, mostly Chenxi, I trained our DCGAN model on my computer rather than Colab in order to train the whole dataset and frequently discussed with him about the DCGAN.

Appendix I - Tables

Table.1. Visually compare the generated results

Attribute	**50 epochs training on full dataset**	**100 epochs training on skrinked dataset**
Hair color/style	Learned that anime characters have different hair colors and represent the style of hair with a higher accuracy.	Learned that anime characters have different hair colors, and even bizarre ones.
Eye size/color	Eye positions are always correct and have uniform sizes in a character. It still has eyes with different colors in a character. Although it seems unnatural, it might actually work to deliver more creative improvisations.	Learned that characters have two eyes at certain positions, with different sizes and color. However, the eyes sometimes are not symmetric, and one eye might be ridiculously larger.
Face contour/orientation	The Shapes of faces are more reasonable, and have more clear edges. It has different face orientations and postures.	Learned that faces can have different shapes. But the lower part of faces are always distorted or blurred.
Mouth	The shape and angle of mouth are more precisely assembled. The models seem to learn about ‘smiling’ and other variations of mouth shape.	Although it kind of learned that faces should have a mouth, it always makes it very blurry and does not seem to understand the correctness of shape and angle.

Appendix II - Figures

Figure 1 - Training on 50 vs 100 epochs

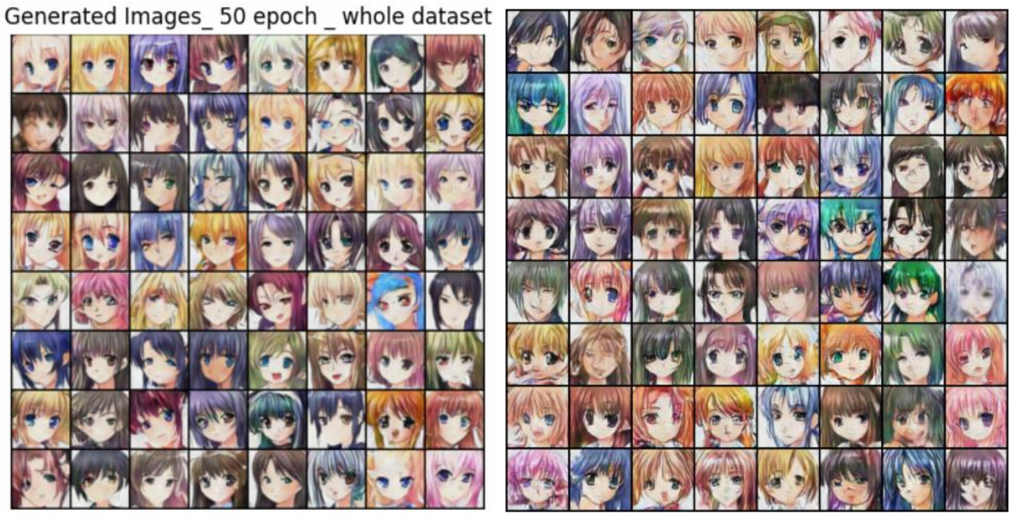


Fig.1. Left: 50 epochs on the full dataset, Right: 100 epochs on the shrinked dataset

Figure 2 - Manipulation of Z representation

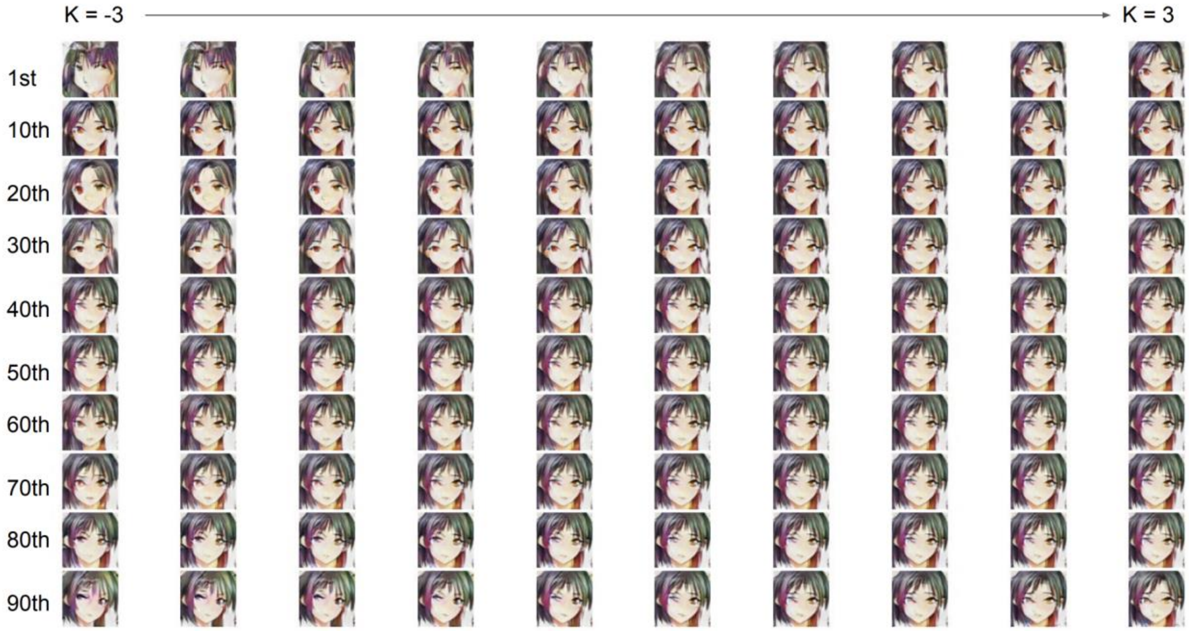


Fig.2. Each row shows the transformation of output images when we adjust that Z variable with the corresponding manipulation factor K.

Figure 3 - Manipulation of Z representation

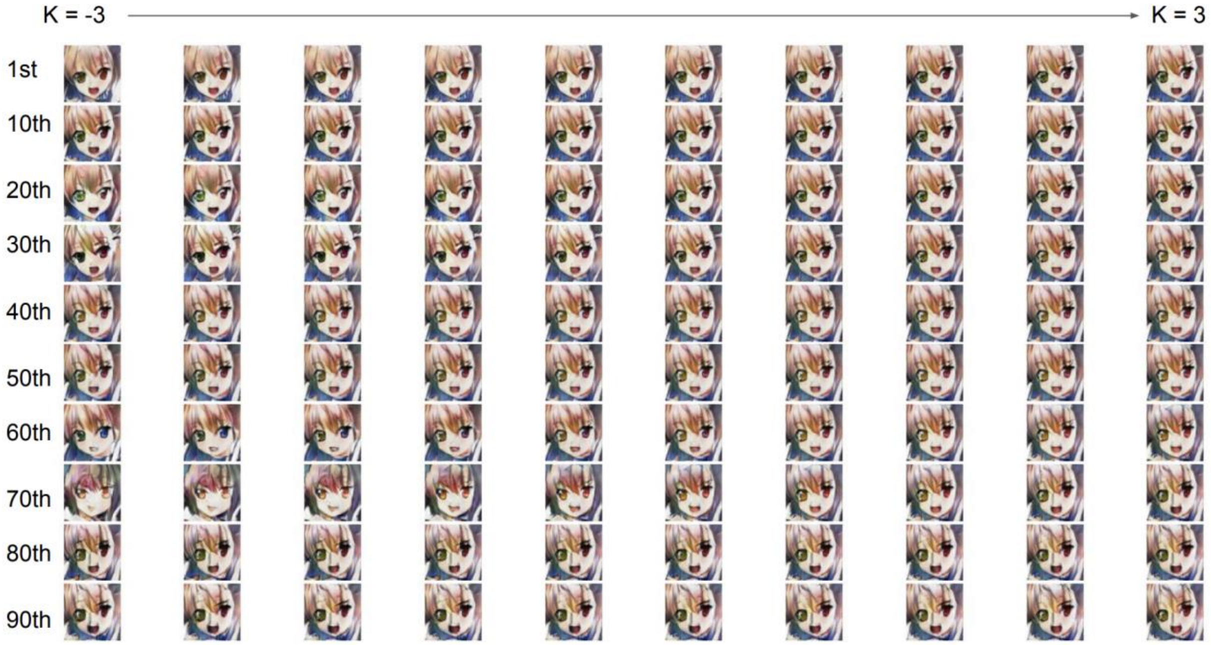


Fig.3. Each row shows the transformation of output images when we adjust that Z variable with the corresponding manipulation factor K.

Figure 4 - DCGAN FID Plot

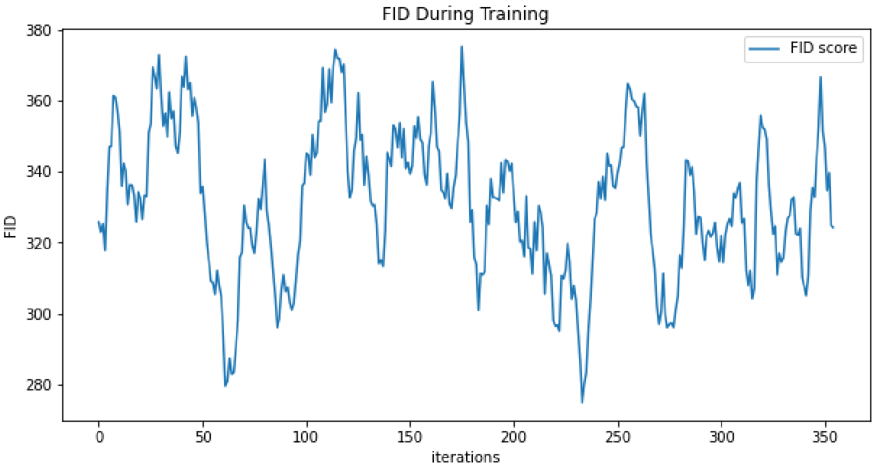


Fig.4. Plot of FID

Figure 5 - DCGAN Loss Plot

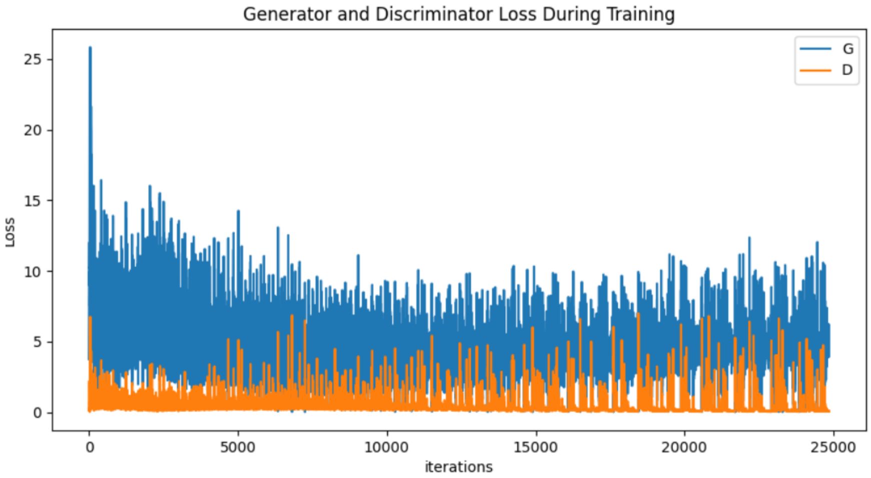


Fig.5. Loss of the Generator and Discriminator

Figure 6 - StyleGAN Training Progression

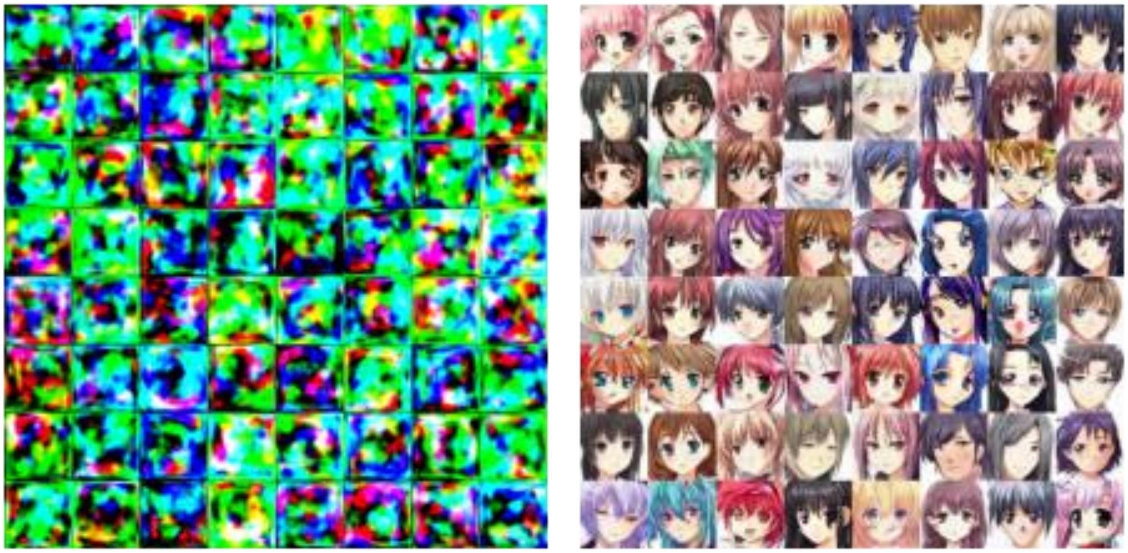


Fig. 6. Left: At genesis. Right: At completion (after 3,500 iterations)

Figure 7 - Style Mixing

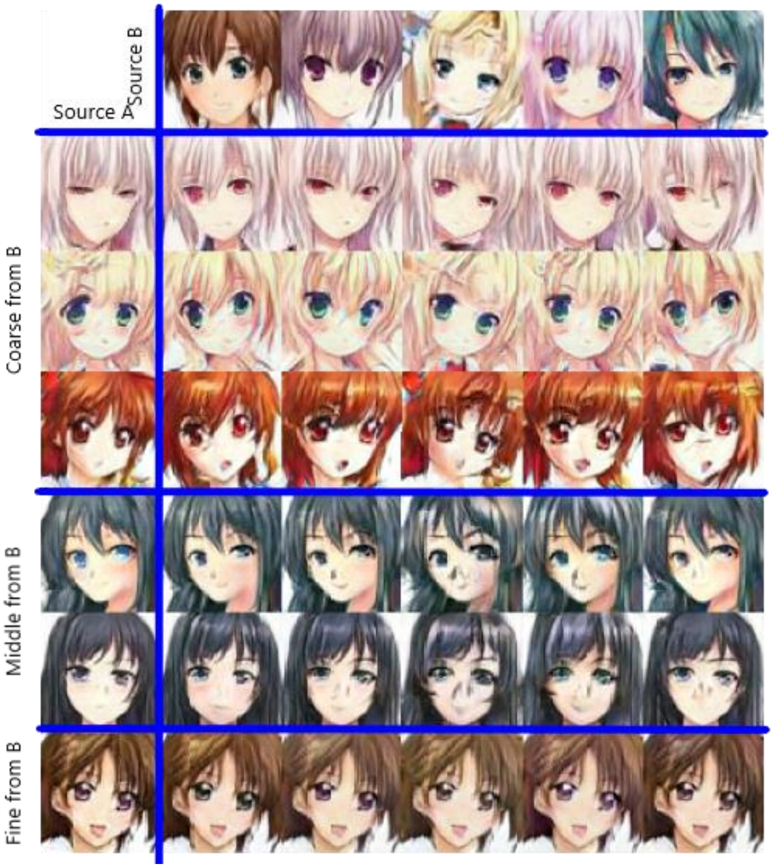


Fig. 7. Styles from source B were combined with the full image of source A.

Figure 8 - Truncation Tricks

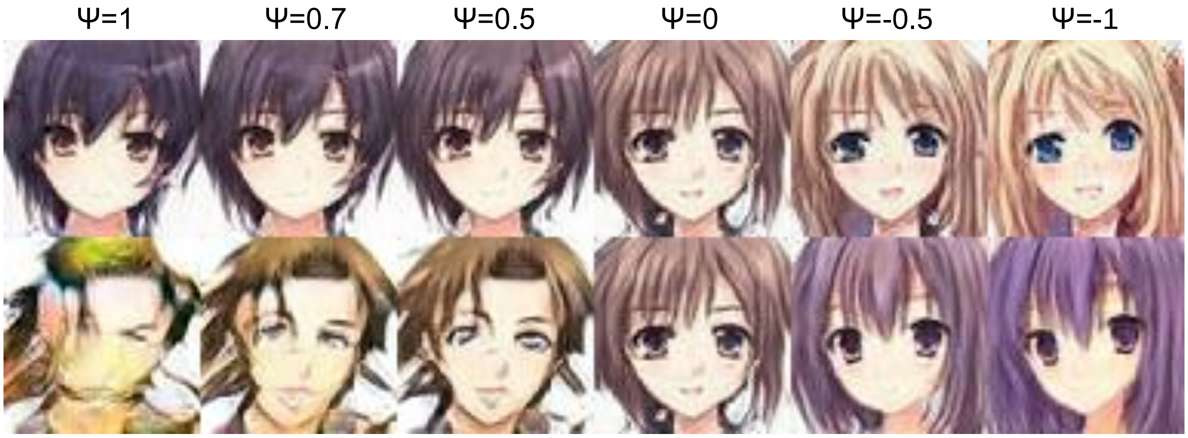


Fig. 8. Truncation Trick

References

1. Goodfellow, et al., Generative Adversarial Networks, https://arxiv.org/abs/1406.2661
2. Churchill, Anime Face Dataset, Kaggle, https://www.kaggle.com/splcher/animefacedataset
3. Radford, Metz and Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, https://arxiv.org/abs/1511.06434
4. DCGAN Tutorial, https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
5. NVLabs, StyleGAN — Official TensorFlow Implementation, GitHub, https://github.com/NVlabs/stylegan
6. NVIDIA – Karras, Laine and Aila, A Style-Based Generator Architecture for Generative Adversarial Networks (2019), https://arxiv.org/abs/1812.04948