Anime Character Creation with GAN

Holly Liang
Electrical Engineering
Stanford, CA
xuejiao@stanford.edu

Leina Sha
Management Science & Engineering
Stanford, CA
renasha@stanford.edu

Abstract

Ever since Generative Adversarial Networks (GAN) was proposed by Goodfellow in 2014, multiple works have concerned themselves with generating synthetic cartoon images ranging from Pokemon to anime faces. However, open-source projects show very blurry and weird pictures. Most, if not all, anime avatar content generation is still manually completed. Our goal is to apply different GAN models to this problem and generate visually pleasing anime faces using GAN.

In this paper we will explore anime character creation using a combination of DC-GAN, W-GAN and spectral/ batch normalization. We will also apply super resolution using SR-GAN to our generated images to further improve image quality.

Our findings indicate that DC-GAN with W-GAN extension yields the best and most stable results. Spectral normalization, in contrast, performed poorly. We conjecture this is due to a low amount of variation in our dataset since 2-D anime faces are quite similar from one to the next with low structural complexity and variation.

1. Introduction

In many scenarios, such as in game design and content creation, graphic artists are needed to draw cartoon or anime avatars. It takes years of schooling and practice for skilled graphics artists to mature. As thus, they are not only expensive to hire but also hard to find. Furthermore, finding an artist that is able to produce a specific style of art that best suits your needs is even more difficult.

If we could instead teach computers how to draw anime characters, then we can decrease the amount of human labor needed and increase the number and diversity of outputs. This would significantly lower the barrier of entry to a large number of endeavors requiring avatar creation. Anyone with access to the model will be able to generate cute and unique avatars for their projects regardless of their drawing aptitude and/ or access to professional artists. Our work will attempt to bring this vision into reality by developing a Generative Adversarial Network (GAN) that will be trained on existing anime avatars and be ultimately capable of generating unique, new avatars indistinguishable from hand drawn ones.

There has been ample work done on anime generation with DC-GAN such as by Jayleicn [1] and Mattya [2]. However, the images produced from these are high noise and low resolution. The highest quality output we found is by Jin et al [3] but they relied on a specialized, clean dataset that is hard to acquire and contains datapoints of similar styles. Also, their model is not open-source, so it is hard to judge the stability of the model. Our goal is to extend current work by generating high quality anime avatars using a generalized, broad dataset. We set out to accomplish this by a) using new and improved models to generate avatars and b) applying super resolution on generated outputs.

Our baseline results using DC-GAN produced satisfactory results that were on par with previous works. The DC-GAN with W-GAN extension model yielded our best and most stable results. Our models that contained spectral normalization layers instead of batch normalization, however, all yielded poor results. This includes DC-GAN with spectral normalization and DC-GAN with W-GAN and spectral normalization. Lastly, our super resolution model using SR-GAN produced mixed results. Although we achieved good outcomes using our training dataset consisting of hand-drawn anime avatars, when we fed in our GAN generated avatars, the output was of poor quality.

2. Related Work

Anime character generation using GAN is a space that has not been deeply explored yet. Related works are sparse and tend to skew towards informal personal projects and experimentation (as opposed to guided research with formal papers). As thus, documentation on data gathering and processing, network architectures and the performance criterion used is lacking or nonexistent. Numerous implementations of anime avatar generation exist, but all except one of the related works we found on the subject were monotonic, uniformly approaching the problem using vanilla DC-GAN.

2.1. DC-GAN Models

The most popular method of anime avatar generation uses vanilla DC-GAN and a highly diverse dataset consisting of tagged anime images from online image boards such as Danbooru [11], Pinterest [12] and Safebooru [13]. One of the earliest examples is Mattya's chainer-DCGAN [2] and Jayleicn's AnimeGAN [1] that launched shortly after the publication of DC-GAN. Following this, many more imitations cropped up. There is only a small variation between these works with most differences attributable to the addition or subtraction of a few layers in the network, hyperparameter settings and output image sizes. The avatars created using this model, however, were uniformly of low quality. The generated images were small in size (most were only 32x32 pixels) and contained significant noise and pixilation with some outputs having unrecognizable facial features.

We used this model as our baseline implementation. We also used the same dataset as Jayleicn. Hence, our baseline model also achieved similar results as these models. Our work diverges from here, however, as we implemented additional models by combining DC-GAN with W-GAN and spectral/ batch normalization. We also performed an extra step of super resolution using SR-GAN on our generated output which these models did not engage in.

2.2. Jin et all [3]

The only paper we found concerned with this subject is Jin et all [3]. Jin approaches anime character generation using a novel, two-pronged method. Their model consists of a modified SRResNet architecture combined with a novel gradient penalty scheme called DRAGAN (Deep Regret Analytic Generative Adversarial Networks. In efforts to reduce noise, they also use a specific, high-quality dataset consisting of character illustrations obtained from a single Japanese ecommerce website selling video games. The avatars in their dataset are all frontal facing with an all-white background. Jin was able to produce aesthetically pleasing, high quality avatars using this method. However, a major disadvantage of their generated outputs is low variation - most of their avatars look very similar to each other.

Our work is different from Jin in both the models and dataset used. We favored Jayleicn's dataset as compared to Jin's as Jin's is overly specialized (i.e. a network trained on this dataset will only be able to produce frontal facing avatars), non-representative (their entire dataset originates from a single source) hard to scale (it will be hard to find large quantities of anime images where the background is pure white and the avatar is frontal facing) and narrow in the artistic styles it encompasses. In addition, there are only 42,000 images in Jin's dataset, our dataset is 3.5x larger.

3. Data

The dataset used consists of ~143,000 images with 126 different attribute tags, featuring anime character faces with diverse facial attributes (hair style, eye color etc), facial orientation and background setting drawn in various artistic styles. The images originated from the Danbooru website [3] and this is the same dataset that was used by Jayleicn [1].

To obtain it we crawled Danbooru using the gallery-dl crawler [10]. Then we processed the data by getting facial crops of these pictures using animeface [9] to yield pictures with a uniform size 96x96 pixels. We also did a manual once-over of the data and deleted bad images that did not contain an anime face or an incomplete face to further reduce noise.

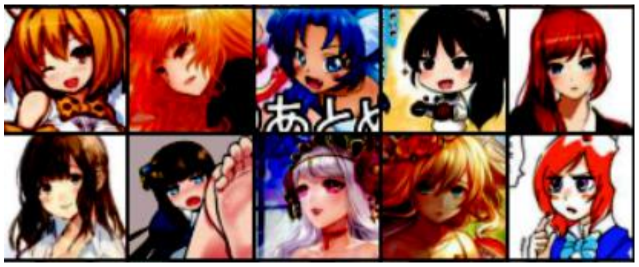


Fig 3 Sample of images from our dataset. Images are diverse in character attributes, pose, background and artistic style.

4. Methods

4.1. Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN) proposed by Goodfellow [4] is a framework wherein two networks, a generator (G) and a discriminator (D), are co-trained to work against each other. Given a target distribution $P_r(x)$, the generator aims to create a model distribution $P_g(x)$ that closely mimics the target distribution, while the discriminator's goal is to distinguish the model from the target distribution. A good analogy is that the two networks engage in a two-player minimax game with the following objective:

$$\min_G \max_D (D, G) = E_{x\ p_{data}}[\log(D(x)] + E_{z\ p_z(z)}[\log(1 - D(G(z)))]$$

By consecutively training the two networks we will strengthen both the generator's ability to create realistic output and the discriminator's ability to discern real output from fake output. Although GAN is notoriously hard to train and very sensitive to a change in hyperparameters, it is currently the most popular generative model for image creation. For our purposes, we will be training different flavours of GAN so as to arrive at a generator network capable of creating high quality anime avatars.

4.2. Deep Convolutional-GAN (DC-GAN)

Deep Convolution GAN (DC-GAN), proposed by Radford et al [5], adapts the GAN framework for the task of image generation by combining it with Convolutional Neural Network (CNN) architecture. Compared to vanilla GAN, DC-GAN enjoys more stable training and higher resolution outputs. It has a few key modifications from GAN including: a) using convolutional layers for the discriminator and transposed convolutional layers for the generator instead of fully connected networks, b) eliminating pooling layers and replacing them with strided convolutions in the discriminator and fractional strided convolutions in the generator and c) using batch normalization between layers in both the generator and discriminator.

We used a modified DC-GAN architecture for our baseline. The generator network has one more layer than the discriminator since our goal is to optimize for generator performance. Fig 3.1 is an image representation of the architecture of our DC-GAN model.

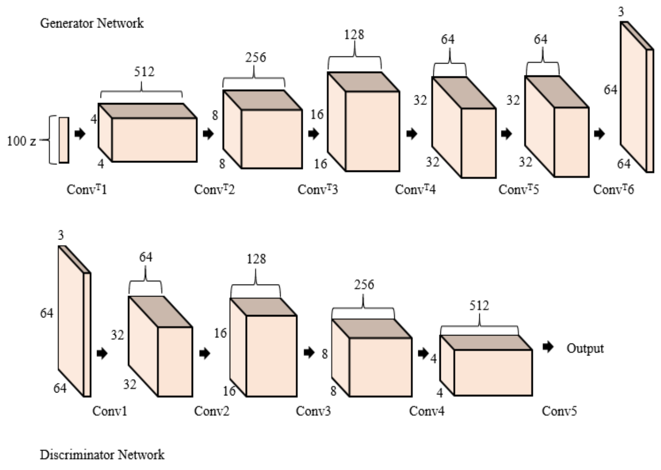


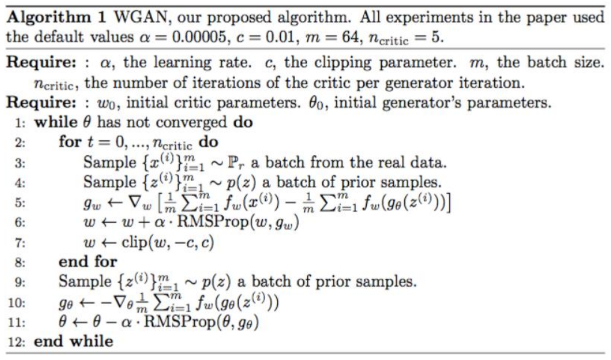
Fig 4.2 DCGAN Architecture

4.3. Wasserstein-GAN (WGAN)

Wasserstein GAN (WGAN), proposed by Ajovsky et al [6], is a modification of GAN that attempts to stabilize training and address the mode collapse problem (where the generator only generates output from a very narrow distribution that only covers a single mode in the data distribution). Generally, GAN models aim to minimize the f-divergence which is a function of the density ratio $P_r(x)/P_g(x)$. However, oftentimes the supports of the two distributions do not overlap significantly and in these areas the density ratio will be zero or infinity. When this occurs, it is highly likely that a discriminator can be produced that can perfectly distinguish between the real and fake inputs. This would lead to the generator getting close to zero gradient – the vanishing gradient problem. WGAN aims to solve this problem by minimizing an approximation of the Earth Mover (EM) distance. EM distance is defined as following:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

In order to use the EM distance a few further modifications to the GAN framework is required. The complete WGAN algorithm is detailed in Fig 3.2 below.



```
Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used
the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.
Require: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
 1: while $\theta$ has not converged do
 2:     for $t = 0, ..., n_{\text{critic}}$ do
 3:         Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
 4:         Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
 5:         $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
 6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
 7:         $w \leftarrow \text{clip}(w, -c, c)$
 8:     end for
 9:     Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
10:     $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:     $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
```

Fig 4.3 WGAN Algorithm

We applied the WGAN algorithm to our DC-GAN architecture. Since the vanilla DC-GAN model generally yielded noisy results, we expected an improvement by stabilizing training using WGAN.

4.4. Spectral Normalization

Spectral Normalization, proposed by Miyato et al [7], is a normalization technique that addresses the same problems as WGAN. By constraining the spectral norm of each layer ($h_{in} \rightarrow h_{out}$), spectral normalization controls the discriminator's Lipschitz constant. σ(A), the spectral norm of matrix A, is equivalent to the largest singular value of matrix A. Or more precisely:

$$\sigma(A) := \max_{\boldsymbol{h}:\boldsymbol{h}\neq\boldsymbol{0}} \frac{\|A\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} = \max_{\|\boldsymbol{h}\|_2 \leq 1} \|A\boldsymbol{h}\|_2,$$

We then normalize the spectral norm of W so that the Lipschitz constraint (σ(W) =1) is satisfied. In particular:

$$\bar{W}_{\text{SN}}(W) := W/\sigma(W).$$

Spectral normalization is a relatively new technique published in Feb 2018. It's claim to fame is that it is the first algorithm to successfully generate samples from all 1,000 ImageNet classes simultaneously. It was found to perform equal or better in image generation in comparison to batch normalization, layer normalization and WGAN on the CIFAR-10 and STL-10 datasets. In addition, it has the added bonus of being sample data independent.

We used spectral normalization in different combinations with DC-GAN and WGAN. Although anime avatars generated from DC-GAN were quite diverse in facial attributes (such as hair, eye color etc), the artistic styles were quite similar across outputs, especially given the diversity of styles in the input dataset. We expected our spectral normalization models to yield more stable results than our DC-GAN baseline with perhaps one or two distinct artistic styles being represented in our output.

4.5. Super Resolution-GAN (SR-GAN)

Super resolution, a highly challenging problem in machine learning, is concerned with transforming a low-resolution image into a high resolution one. Over the years there have been various methods proposed for solving this problem ranging from non-deep learning algorithms to approaches using CNN. Super Resolution GAN (SR-GAN), published by Ledig et al [8], is commonly viewed as state of the art super-resolution today and is the method that we will be using. We shall apply super resolution on our generated anime avatars to improve their quality by decreasing noise and pixilation.

The architecture employed by SR-GAN is a deep residual network (ResNet) with skip connections. It is capable of generating photo-realistic reconstructions of low-resolution images for up to 4x upscaling factor. The architecture used by SR-GAN can be seen in Fig 4.5.

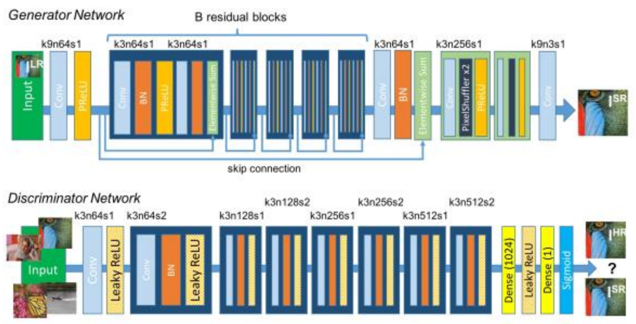


Fig 4.5 SR-GAN Architecture

SR-GAN also uses a new perceptual loss function that takes into account content loss and adversarial loss. Here the loss is calculated based on the VGG network's feature maps, a significant deviation from MSE loss used before. In particular:

$$l^{SR} = \underbrace{\underbrace{l_X^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}}_{\text{perceptual loss (for VGG based content losses)}}$$

$$l_{VGG/i.j}^{SR} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

To train our SR-GAN we resized our dataset of cropped anime faces to 64x64 and used these as ground truth images. Then we down sampled these by 2x to obtain the low-resolution images.

5. Experiments

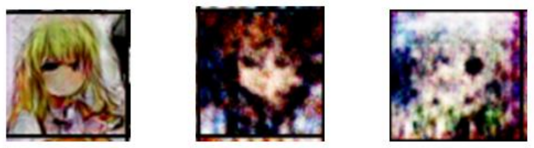
5.1. DC-GAN

We trained our DC-GAN model with the full dataset (~143,000 images) over 200 epochs. We used a binary cross entropy loss function with mini-batch gradient descent with batch size 128, output image size 64x64, and latent vector z of length 100. To obtain the best results we experimented with using different optimizers, network architectures and learning rates. Table 5.1.1 below includes a summary of our experiments.

Variable	Experimentation
Optimizer	• Adam (beta1 = 0.5, beta2 = 0.999) • SGD
Generator Conv Layers	• 6 • 7
Learning Rate	• 0.0025 • 0.002 • 0.0015

Table 5.1.1 DC-GAN Experiments

For many of these experiments it was easy to tell that the results would be poor after a few epochs of training and we stopped there. The best parameters were using Adam, 6 layers of convolutional layers for the generator and a learning rate of 0.002. We conjecture that 6 layers of convolutional layers are better than 7 because at 7 layers the generator is much more robust than the discriminator causing the discriminator to perform poorly. If the discriminator cannot adequately discern real from fake outputs the generator will also be unable to learn how to create more realistic fakes. Fig 5.1.2 compares outputs for across different experiments after 10 epochs of training.



Best Params *Using SGD* *7 Conv Layers*

Fig 5.1.2 Experiment Outputs after 10 Epochs

Fig 5.1.3 below are the results we obtained from training with our best parameters. As you can see, almost all of the generated avatars have recognizable facial features, with hair and eyes being the most prominent. There are a couple of high quality images (~20%) where the avatar created is atheistically pleasing with smooth line and colour transitions (boxed in green). There are also a couple (~14%) of low quality images where the facial features are unrecognizable (boxed in red). There is a significant amount of blurring and noise distortion in all outputs as we expected. The avatars generated are quite diverse, with none looking very similar to another. These results were better than the average DC-GAN output mentioned in related works as we had a higher percentage of high quality images (~20%).

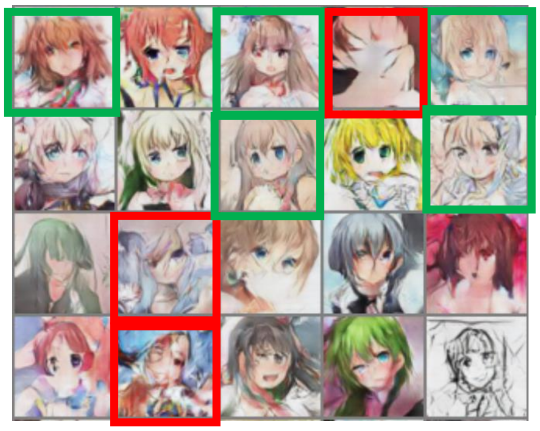


Fig 5.1.3 DC-GAN output. High quality outputs are boxed in green, low quality in red.

Fig 5.1.4 below details our generator and discriminator training loss over 200 epochs. Although the variance is quite high the general trend is towards convergence for both networks.

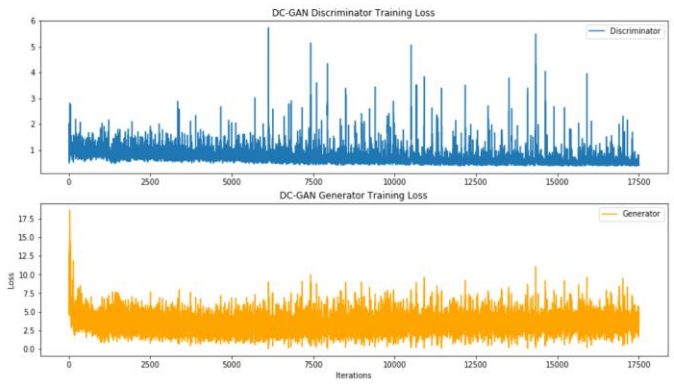


Fig 5.1.4 DC-GAN Training Loss

5.2. DC-GAN + Spectral Normalization

DC-GAN with spectral normalization yielded poor results overall. We trained for 200 epochs over the full dataset with a binary cross entropy loss function with mini-batch gradient descent with batch size 128. We increased the number of epochs as we were getting poor results and hypothesized that the network needed more training in order to converge. We also experimented with different optimization parameters, network architectures, output sizes and learning rate decay. We also implemented a model combining DC-GAN, WGAN and spectral normalization but did not see significant improvements in our output. Table 5.2.1 below includes a summary of our experiments.

Variable	Experimentation
Adam Parameters	• **Beta 1 = 0** • Beta 1 = 0.5 • **Beta 2 = 0.9** • Beta 2 = 0.999
Conv Layers	• G: 6 conv, D: 5 conv • G: 5 conv, D: 5 conv • G: 5 conv, D: 6 conv • **G: 5 conv, D: 8 conv**
Output Size	• 64x64 • **32x32**
Learning Rate Decay	• **No decay** • Decay from 0.002 to 0.0002

Table 5.2.1 Spectral Normalization Experiments

None of these significantly improved our results except for decreasing the output size from 64x64 to 32x32. Fig 5.2.2 below compares the 64x64 outputs to the 32x32 ones.

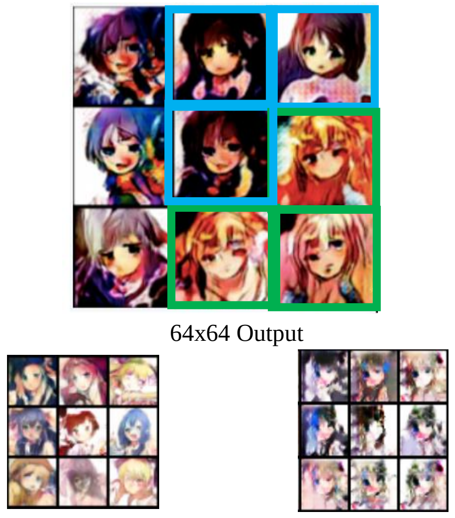


Fig 5.2.2 Spectral Normalization Output

As you can see, for the 64x64 outputs, the facial features are barely recognizable, there is significant distortion and noise in the generated images and the output is not diverse. Most of the images appear to have the same underlying facial structure (similarly shaped hair and face) with varying colorizations. Images that are extremely similar to each other are boxed in the same colors in Fig 5.2.2.

For the 32x32 outputs however, we can see that performance hit its peak at around 30 epochs of training. At ~30 epochs the avatars have distinguishable facial features and look quite aesthetically pleasing. More importantly, there is great diversity in outputs. After 30 epochs however, performance starts to worsen. At around 120 epochs there is significant noise and blurring in the image and almost all of the avatars have similar underlying structure.

Although the 32x32 outputs at 30 epochs looked more aesthetically pleasing at first glance, this might not actually be because of an improvement in quality. Since the avatars contain less pixels it is easier for the network to train. In addition, as humans, we have a tendency to fit and interpret images using our mental frameworks. This may cause us to look at a low-resolution image, fill in the blanks, and imagine a beautiful high-quality version of it when in actuality the high-resolution version could be completely different.

For both 64x64 and 32x32 outputs however, we notice that the avatar colorings are much brighter and more sophisticatedly rendered. While the DC-GAN outputs had dull colors almost resembling hand drawn images, the spectral normalization coloring looked digitally rendered.

We were very surprised by our results because they were the exact opposite of what we expected. Spectral normalization is designed to produce more diverse outputs but our outputs tended towards more similar outputs after more epochs of training. We conjecture this is because spectral normalization performs well when producing different *classes* of images such as horse, car etc, where one class is significantly different from each other in terms of image structure (i.e. a horse has four legs, a car is rectangular shaped). For anime faces however, the overall structure of the image is pretty similar, and so spectral normalization has a hard time finding different classes. As thus, after extensive training, it only generates images from one or two classes which is why there are a lot of monotonic images. In addition, the more vibrant colors in spectral normalization is possibly due to the network classifying black and white images of one class and multicolored ones of another which makes the output's colors deeper instead of the pale, pencil-colored effect we get from DC-GAN.

Fig 5.2.3 below details our generator and discriminator training loss over 200 epochs. From the graph we can see that both networks converged quickly and efficiently.

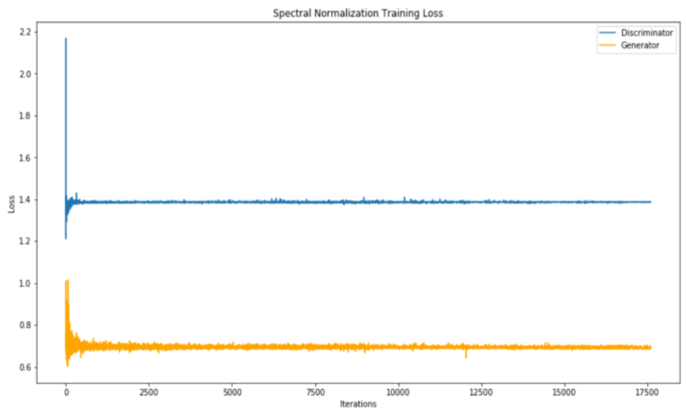


Fig 5.2.3 DC-GAN + Spectral Normalization Training Loss

5.3. DCGAN + WGAN

DCGAN + WGAN yielded our best results. We trained for 200 epochs over the full dataset with the Wasserstein loss function and mini-batch gradient descent with batch size 128. We used a learning rate of 0.002 with an Adam optimizer having beta1 = 0.5 and beta2 = 0.999. Fig 5.3.1 below are the results we obtained.

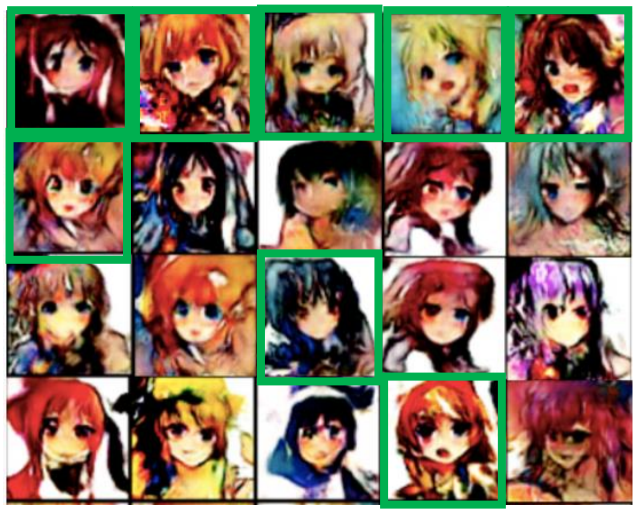


Fig 5.3.1 DC-GAN + WGAN Output

As you can see, **all** of the generated avatars have recognizable facial features. Compared to DC-GAN, there are a lot more high-quality images (~40%) where the avatar created is atheistically pleasing with smooth line and colour transitions (boxed in green). There are **no** low-quality images with unrecognizable facial features. Although there are one or two similar anime faces the output is still sufficiently diverse. In addition, the colours are also deeper and more vibrant. However, there is still significant noise present in the images. Overall, the output is more stable.

Fig 5.3.2 below details our generator and discriminator training loss over 200 epochs. Although the loss seems to explode around 2000 iterations, the overall trend is converging for both networks.

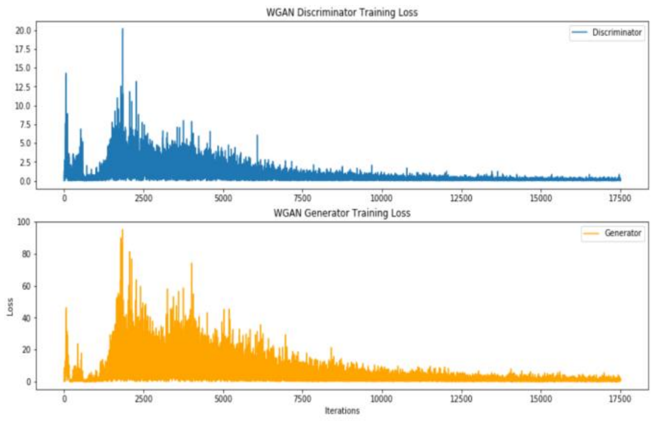
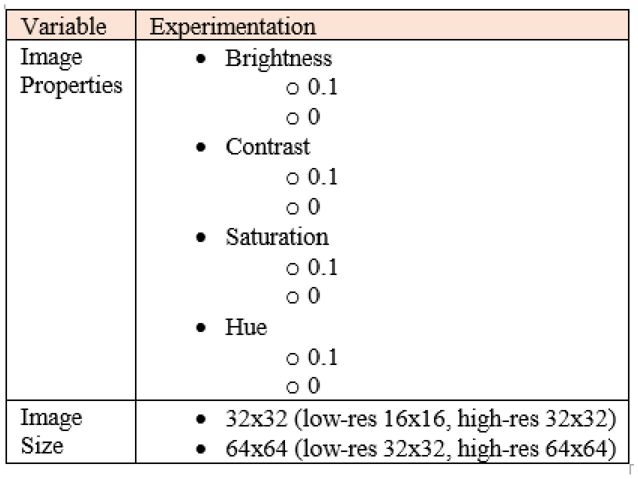


Fig 5.3.2 DC-GAN + WGAN Training Loss

5.4. SR-GAN

SR-GAN gave us mixed results. While it performed well on the training dataset (our original dataset from Danbooru), it performed poorly when we inputted images generated from our GAN models. We trained for 200 epochs over the full dataset with mini-batch gradient descent and batch size 128. We used a learning rate of 0.0001 with an Adam optimizer. Our original dataset down sampled by 2x was our ground truth image and the original dataset down sampled by 4x was our low-resolution input.

We hypothesized that the reason our model performed poorly on GAN generated images is that there is significantly more noise and irregularity in these images. Thus, we tampered with our ground truth images in an effort to increase their noise levels to be on par with the generated avatars. We experimented with pre-processing our data by applying different brightness, contrast, saturation and hue. We also tried different down sampling ratios and output sizes. Table 5.4.1 below includes a summary of our experiments.



Variable	Experimentation
Image Properties	• Brightness ○ 0.1 ○ 0 • Contrast ○ 0.1 ○ 0 • Saturation ○ 0.1 ○ 0 • Hue ○ 0.1 ○ 0
Image Size	• 32x32 (low-res 16x16, high-res 32x32) • 64x64 (low-res 32x32, high-res 64x64)

Table 5.4.1 SR-GAN Experiments

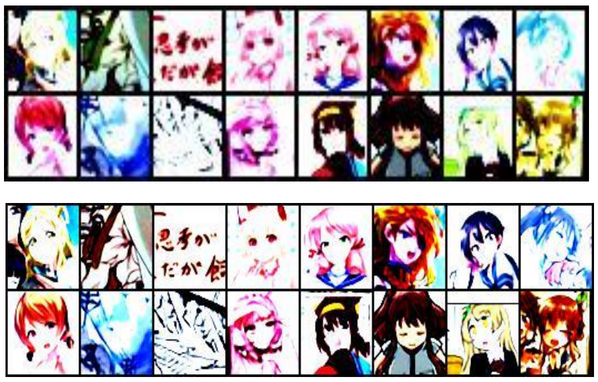


Fig 5.4.2 SR-GAN Training Output on Real Images

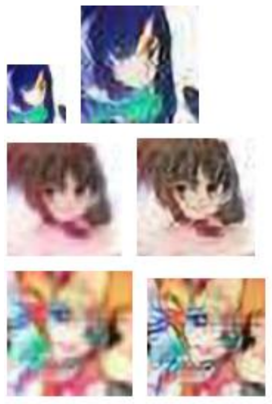


Table 5.4.3 SR-GAN Test Output on GAN Generated Images

Varying output size yielded no material changes. Training on a dataset with higher noise (as a result of tuning up brightness, saturation etc) however, decreased the model's training performance against real anime images but increased its performance at super resolution of our GAN generated images. Fig 5.4.1 and Fig 5.4.2 details our training and test results.

Fig 5.4.5 below details our generator and discriminator training loss over 200 epochs. Although our generator loss remained high, our discriminator loss converged.

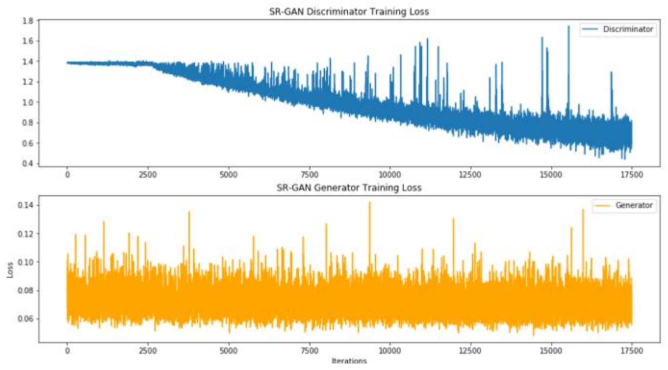


Table 5.4.5 SR-GAN Training Loss

6. Conclusion

The best model appears to be DC-GAN with WGAN, it achieved consistent high-quality outputs and produced little to no indistinguishable faces. From this project, we learned that GANs are extremely hard to train, discriminator to generator layers ratio can often influence the result a lot. The network also tends to be unstable, and it can collapse due to any known/ unknown reasons. More often than not, what looks better to GANs might be different from what looks better to us. Even though the loss on D and G converged, the result can still look abnormal to us.

Having a good dataset is very important. Our results suffered a lot from the noisy dataset. When we looked into the dataset, we found out that almost 10% of the dataset do not contain a clear face. Faces are also angled/tilted or just a side profile. The background is noisy too. This might be the key reason why our generated images are low resolution, and it could be a cause of instability. Moving forward, we are definitely going to fix this problem by either find a better source of data, e.g. game character introduction page where all characters faces forward and background is one color, or manually prune the dataset.

We also learned about the limitations of different networks. For example, SR-gan learns the transition from a blurry image to a high-res image, however, our gans-generated images are blurry in a different way than just low-res and jitter. Thus direct application is not a good idea. Also, spectral normalization is good for generating multiple different classes of images, but not particularly good for generating one kind of images. That's why we see groups of pictures that are similar to each other. In the future, we can try different network structures such as conditional GAN.

References

[1] Jayleicn, AnimeGAN, 2017, https://github.com/jayleicn/animeGAN

[2] Mattya, chainer-DCGAN, 2016, https://github.com/mattya/chainer-DCGAN

[3] Jin et al, Towards the High-quality Anime Characters Generation with Generative Adversarial Networks, 2017, https://arxiv.org/abs/1708.05509

[4] Goodfellow et al, Generative Adversarial Nets, 2014 https://arxiv.org/pdf/1406.2661.pdf

[5] Radford et al, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2016, https://arxiv.org/pdf/1511.06434.pdf

[6] Arjovsky et al, Wasserstein GAN, 2017, https://arxiv.org/pdf/1701.07875.pdf

[7] Miyato et al, Spectral Normalization for Generative Adversarial Networks, 2018, https://arxiv.org/pdf/1802.05957.pdf

[8] Ledig et al, Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, 2016, https://arxiv.org/pdf/1609.04802.pdf

[9] nya3jp, python-animeface, 2017, https://github.com/nya3jp/python-animeface

[10] mikf, gallery-dl, 2018, https://github.com/mikf/gallery-dl

[11] Danbooru, danbooru.donmai.us

[12] Pinterest, https://www.pinterest.com/

[13] Safebooru, https://safebooru.org/