GANime: Image-to-Anime Translation using Generative Adversarial Networks

Wesley Liao
Georgia Institute of Technology
wliao63@gatech.edu

Gary Liu
Georgia Institute of Technology
gliu331@gatech.edu

Nikki Hu
Georgia Institute of Technology
lhu81@gatech.edu

Abstract

One of the most popular applications of Generative Adversarial Networks is to do image-to-image translation. This task entails taking an image from one domain and translating it into another domain while maintaining some level of semantic meaning. In this report we explore different methodologies for this task and perform experiments with translating portraits of real people to Anime or cartoon-like character faces. We experimented with 3 different approaches for this task: AnimeGAN, U-GAT-IT, and MUNIT. We found that AnimeGAN, a neural-style transfer based architecture, did not perform well in this setting due to difficulty in capturing the true distribution of the target domain and finding appropriate weights for each loss function. CycleGAN based approaches (the latter two) performed quite well and were able to generate reasonable translations.

1. Introduction/Background/Motivation

The main task that we focused on was portrait-to-Anime translation. The problem is as follows: given an image of a human face, can we translate that image into a picture of an Anime character where there is some similarity between them, in terms of hair style, expression, head orientation, or general style.

With increasing use of social media and photo sharing in the last decade, image filters on platforms such as Instagram and Snapchat have exploded in popularity. One such popular filter is an anime face filter where a neural network is used to translate a person's face into an anime face. With high quality translations, social media filters will greatly improve.

Another application of Anime generation is in the animated film making industry. Animators currently draw the majority of the frames shown in a show or movie. Recently there has been advances in using Deep Learning to automate some of this process. One approach is to use the translation approach, whether from real images, scripts, or even descriptions of characters. The other is frame interpolation [16], which is when coarsely animated sequences are fed to a network for producing filler frames between each of the human drawn frames.

2. Related work

2.1. CartoonGAN

CartoonGAN [4] was first introduced to address the problem of photo transformations of real world scenes to cartoon/anime styles by implementing neural style transfer under a GAN framework where unpaired photos and cartoon images were trained. The goal of the network was to learn a mapping between photo and cartoon manifolds that retains the real world photo's semantic content while adapting to the anime image's artistic style. To accomplish this, the authors proposed two novel losses: (1) a semantic content loss ($\mathcal{L}_{con}$) which is calculated as the l_1 loss of the high-level feature maps between the real world photo and anime image, as extracted by a pre-trained VGG19 network, and (2) an edge-promoting adversarial loss ($\mathcal{L}_{adv\_edge}$) for preserving clear edges because anime images tend to be have highly simplified textures, clear edges and smooth color shadings. The paper also proposed an initialization phase to pre-train the generator network. By pre-training the generator to minimize the content loss between the input image and the generated image, the authors were able to improve the convergence of the network and prevent the model from being trapped at sub-optimal local minimums.

CartoonGAN's generator network starts off with a flat convolution layer (conv, norm, relu) followed by two down-convolution blocks that spatially compresses and encodes the images while extracting local features to be used for downstream transformations. Next, eight residual blocks

of identical composition as proposed by Johnson et el [8] are used to construct features between the content of the photo and the style of the cartoon image. Residual networks were used here to address the vanishing gradient problem. Lastly, the extracted features are fed into two up-convolution layers followed by a final convolution layer to reconstruct the cartoonized image. To complement the generator a shallow discriminator network is used to quickly judge whether an image is cartoon or not. The discriminator network is composed of a flat convolution layer followed by two strided convolutional blocks for feature extraction. The local features are then passed to a feature construction block followed by a 3 x 3 convolutional layer for classification. Leaky ReLu is used throughout the discriminator network to avoid the "dying ReLU" problem.

2.2. AnimeGAN

AnimeGAN [2] improved upon the results of CartoonGAN by introducing three more loss functions calculated between the real world photo and the anime image: (1) grayscale style loss ($\mathcal{L}_{style}$) which measures the l_1 loss between the gram matrix of the high-level semantic features extracted by VGG19 (grayscale was used so the feature extract will focus even more on style and texture and less on color), (2) color reconstruction loss ($\mathcal{L}_{color}$) which measures the l_1 loss of each of the YUV color channels and (3) grayscale adversarial loss ($\mathcal{L}_{adv\_gray}$) which is used to counteract the coloring affects of the grayscale style loss and prevent the generated image from being displayed as grayscale. The final loss for AnimeGAN for the generator and discriminator is as follows:

$$\mathcal{L}_G(G,D) = w_{adv}\mathcal{L}(G,D) + w_{con}\mathcal{L}_{con}(G,D) + w_{style}\mathcal{L}_{style}(G,D) + w_{color}\mathcal{L}_{color}(G,D)$$

$$\mathcal{L}_D(G,D) = w_{adv}\mathcal{L}(G,D) + w_{adv\_gray}\mathcal{L}_{adv\_gray}(G,D) + w_{adv\_edge}\mathcal{L}_{adv\_edge}(G,D)$$

Combining the above three losses along with the losses introduced by CartoonGAN, images generated by AnimeGAN were able to not only preserved the original image's content but also accurately retain its original color and tone while still capturing the vividness and simplistic textures of anime images.

AnimeGAN's generator network follows a similar architecture as that of CartoonGAN with a few modifications: (1) the down-conv and up-conv layers are replaced with normal convolution blocks with bigger kernels, padding and strides to improve the quality of the generated images, (2) inverted residual blocks are used instead of the standard residual blocks which helped to significantly reduce the number of parameters and (3) Leaky ReLU is used instead of Relu to assist with gradient flow. AnimeGAN implemented the same lightweight discriminator network as that of CartoonGAN.

2.3. CycleGAN

One of the major advances in image-to-image translation was CycleGAN[17] and the idea that instead of using paired images to train the generator and discriminator, we train two sets of generators to not only do the forward translation but also backwards as well. We then add an extra loss term called cycle consistency loss to evaluate the quality of the back-translation. This is similar to how auto-encoders have a reconstruction loss to assess how similar the input is to the output. If G is the forward generator and F is the backward generator then the loss is expressed as:

$$\mathcal{L}_{cyc} = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x)-x\|_1] + \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y)-y\|_1]$$

The main benefit of this new formulation is that we no longer need paired image (i.e. labeled data for supervised learning), and instead we can use two sets of images from both domains without any linkage between the data points.

2.4. U-GAT-IT

More recent work builds upon CycleGAN with additional layer types, training schemes, and loss functions. U-GAT-IT[11] is one such model and has achieved impressive results especially in the image-to-Anime translation domain. It does so by adding attention modules to the network and a new normalization technique called Adaptive Layer-Instance Normalization (AdaLIN). The AdaLIN is an improvement on previous types of layer norms, which has previously shown to stabilize the GAN training process. The formulation is as follows:

$$AdaLIN(a,\gamma,\beta) = \gamma \cdot (\rho\hat{a_I} + (1-\rho)\cdot\hat{a_L}) + \beta$$

The $\hat{a_I}$ term refers to the normalized version of a across channels and $\hat{a_L}$ is a normalized across the layer. The γ and β parameters are produced from the generator's encoder. In terms of loss functions, it uses a combination of:

1. Least squares adversarial loss, which is a variant of the standard loss for GANs
2. Cycle loss to assess cycle consistency, taken from CycleGAN
3. Identity loss to evaluate the color similarity between the input image and the output image
4. CAM loss which uses auxiliary classifiers to identify areas in the image where the target domain discriminator needs to look to improve / to discriminate between the domains

The combination of these advancements has resulted in very compelling translations.

2.5. MUNIT

Multimodal Unsupervised Image to Image Translation (MUNIT) [6] is another variable of GAN that takes a slightly different approach to image translation. For the encoder piece, it uses two blocks of encoders to break down the image into content code using the content encoder and style code using the style encoder. This is applied to both the target and the source domain (2 auto-encoders for each domain). For the decoder, MUNIT stacks a residual block and an upsampling block. The residual block combines the content and style code by taking the content code in the residual layers and using affine transformation parameters from the style code in normalization. Adaptive Instance Normalization (AdaIN) is applied, where the transformation parameters are from the style codes:

$$AdaIN(z, \gamma, \beta) = \gamma \cdot (\frac{z - \mu(z)}{\sigma(z)}) + \beta$$

The z term refers to the activation of the previous convolutional layer, and μ and σ refer to the channel-wise mean and standard deviation. The γ and β are generated using a MLP from the style codes dynamically.

MUNIT aims to solve the issue of output diversity as the problem of image to image translation is multimodal in nature, but have often been thought of as a deterministic or unimodal problem. The MLP generates dynamic AdaIN parameters that can produce more diverse outputs.

Total loss for MUNIT is defined as a weighted average of the sum of the bidirectional reconstruction loss and adversarial loss:

$$\min_{E_1,E_2,G_1,G_2} \max_{D_1,D_2} L(E_1, E_2, G_1, G_2, D_1, D_2) = L^{x_1}_{GAN} + L^{x_2}_{GAN} + \lambda_x(L^{x_1}_{recon} + L^{x_2}_{recon}) + \lambda_c(L^{c_1}_{recon} + L^{c_2}_{recon}) + \lambda_s(L^{s_1}_{recon} + L^{s_2}_{recon})$$

The terms λ_x, λ_c, λ_s are used to control the importance of reconstruction terms. Bidirectional reconstruction loss is used in addition to the usual adversarial loss to ensure the learned encoders and decoders are inverse.

2.6. AniGAN

AniGAN[13], also named Style-Guided Generative Adversarial Network is a novel approach that ties together a few other approaches as it tries to improve upon previous issues around artifacts and distortions in the generated images. It is also unsupervised. It starts off with a double encoder architecture similar to MUNIT. For the decoder, it introduces two new types of blocks: ASC Blocks in place of the residual blocks, and FST blocks in place of the upsampling block. It also introduces two new normalizations: point-wise layer instance normalization (PoLIN) and adaptive point-wise layer instance normalization (AdaPoLIN), which improves upon the AdaLIN used in U-GAT-IT.

$$\mathcal{A}daPoLIN(z, \gamma_s, \beta_s) = \gamma \cdot Conv([\frac{z - \mu_I(z)}{\sigma_L(z)}], [\frac{z - \mu_L(z)}{\sigma_L(z)}]) + \beta_s$$

where the expression $Conv()$ denotes a 1 × 1 convolution operation,$[\cdot, \cdot]$ denotes channel-wise concatenation, z is the the feature map of a network layer, μ_I , μ_L and σ_I, σ_L denote the channel-wise and layer-wise means and standard deviations, respectively. PoLIN is implemented similarly to AdaPoLIN but without the transformation parameters. Another novel idea in AniGAN is a double branch discriminator, which aims to address the issue with distorted facial parts and artifacts after translation. This is done through a block of shared shallow layers, and separate deep layers for the two domain specific task (anime to real face and real to anime face). This allows the discriminator to retain domain-shared distributions in the shared layers, while also learning domain specific distributions in the separate layers. In terms of loss function, total loss is defined as:

$$\mathcal{L}_G = L_{adv} + \lambda_{rec} \cdot L_{recon} + \lambda_{fm} \cdot (L_{fm} + L_{dfm})$$

$$\mathcal{L}_D = -L_{adv}$$

The terms λ_{rec}, λ_{fm} are hyper-parameters used to control the importance of the losses. In addition to the usual adversarial loss, AniGAN also uses reconstruction loss, feature matching loss, and domain aware feature matching loss.

The paper was trained on only female images.

3. Data

The Flickr Faces High Quality (FFHQ) dataset [10] was used for human portraits.This dataset consists of 70k images crawled from Flickr, all of which are human faces and have been auto-aligned and cropped. The dataset's photos are very high quality (1024 by 1024 resolution) which is one of the main reasons this dataset was used. The other two reasons being 1) the sheer size of the dataset, and 2) the diversity of facial expressions and face accessories. However, during model development the size of the dataset proved unmanageable so 20k images were used.

The Danbooru2019 portraits dataset [1] was used for anime faces. This dataset is a subset of the greater Danbooru dataset which consists of 4.2 million Anime images scraped from an online message board. The portraits dataset contains 303k images of Anime faces cropped and transformed into 512 by 512 resolution. Upon examining of some of the images, it's clear that not all the images are face-only portraits; there are a number of non-centered

body-included images, and even some that look like comic book panels. This data limitation likely negatively impacted the quality of the models and their outputs as well. Another limitation of this dataset that became apparent is that it is highly imbalanced, containing mostly female characters and does not reflect the same gender diversity as the FFHQ dataset.

Lastly, additional supplemental dataset of unpaired images of real world scenes and anime style scenes was used for training AnimeGAN. The dataset was provided by the AnimeGAN author's repo [3].

4. Approach

To tackle the problem of image translation between real world faces and anime faces, we tried three different GAN approaches in increasing order of success.

4.1. AnimeGAN

We first implemented AnimeGAN by transcribing the authors' tensorflow training code found on their repo [3] into PyTorch. The PyTorch generator code was adapted from [12]. The goal here is to recreate AnimeGAN results on real world photos by training on their provided dataset of animation images from the movie "Your Name" to capture director Miyazaki Shinkai's unique style.

Next we will test the model by running the trained generator on human portraits to see how effective the network is able to translate Shinkai's artistic style to faces. We anticipate the network would perform poorly on translating real world faces to their anime counterparts because the training data is mostly composed of real world scenes such as buildings, roads and nature which have very similar anime counterparts. Anime faces on the other hand are quite different from their human counterparts because they are characterized by their sharp edges, pointed features and simplistic representations.

To address this, we will attempt to tune the hyperparameters to place more emphasis on specific losses such as the grayscale edge promoting loss to help the model learn more specific features of faces.

4.2. MUNIT

We followed the official implementation of MUNIT [5] first as a template for our implementation. We wanted to use it as a baseline to compare to against further variations.

A variation of MUNIT was implemented that included the encoder and decoder architecture proposed by AniGAN while retaining the rest of the architecture of MUNIT (decoder architecture as well as loss functions). Since the two methods also shared similar structure for the encoder piece, we left the encoder architecture as similar as possible to MUNIT. We only changed the decoder architecture and normalization. Specifically, instead of the residual and upsampling blocks used in MUNIT, we swapped in the ASC and FST blocks from AniGAN, and instead of AdaIN we used AdaPoLIN from AniGAN. We wanted to find out whether AniGAN's proposed encoder decoder architecture, which is supposed to have a larger receptive field was enough to produce meaningful changes to the MUNIT results.

4.3. AniGAN

There were no official implementations of AniGAN as the report is fairly recent. We tried to recreate the AniGAN method by using the official implementation of MUNIT as a baseline while also referencing a unofficial implementation of AniGAN [7].

The model was trained and tested on the same dataset as the other methods in order to keep the comparisons fair, even though the paper used different datasets. The paper used a cleaner version of the data where only female faces and non monochrome faces were used. This is likely due to the differences between male and female faces and the added challenges when generalizing to male faces.

To replicate AniGAN, in addition to prior changes made to MUNIT, we also added a double branch discriminator with shared shallow layers and additional loss functions from the paper. We tried different optimizers as the paper used RSMProp and MUNIT used Adam. Due to limited time capacity, we did not include some other details like stablizing training with the hinge version of GAN as GAN loss and adopting real gradient regularization, or using a historical average version of the intermediate generators as the final generator.

We weren't able to generate great results from AniGAN because of the significant variations and parameter turnings needed to fill in the details the paper left out. This would be a great next step to further fine tune the AniGAN model.

4.4. U-GAT-IT

We first attempted to implement our own version of U-GAT-IT. However, the model architecture proved to be more complex and time-consuming than we had bandwidth for, so we instead used took the official PyTorch implementation [9] as a template for our experiments. We modified some of the network architecture to work with our datasets and hardware constraints. We primarily experimented with the four components of the loss function and tested a number of combinations before settling on one set of hyperparameters to train the final model with.

When running with the default configuration, we ran into GPU memory issues even with a batch size of 1. The model itself could not fit in the Tesla V100 GPU. We noticed that the γ, β component of the generator accounted for a significant share of the parameters as it contained 3 fully-connected layers. The purpose of these layers is to generate

the scale and shift parameters for the AdaLIN calculation. By reducing the input dimension of the first MLP layer by a factor exponential to the downsampling ratio, the model was able to fit into memory.

In terms of experimentation, the main focus was on the relative weights of each of the 4 components of loss. We started with the default parameters in the paper and then adjusted each one for 1,000 iterations with batch size of 1 and compared the results qualitatively. We did not adjust the architecture of the generators or discriminators.

5. Experiments and Results

To measure the success of each network we looked at qualitative comparisons of the generated images of each GAN model with the original human portrait. Model performance can be quite obvious because anime artistic styles are highly recognizable and human vision can easily handle the content comparison. Next, for quantitative measures, we looked at Fréchet inception distance (FID) which is a metric used to compare the distribution of the generated image with that of the training image. We used a PyTorch implementation of FID [15] to perform the analysis on generated images. It uses an Inception model trained on ImageNet to calculate the covariance of the activations when both real Anime images and generated ones are fed to the network. It can be expressed as the following:

$$\text{FID} = |\mu - \mu_w|^2 + \text{tr}(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w)^{1/2})$$

Where μ and Σ are the means and covariances of the datasets, respectively.

5.1. AnimeGAN

Recreating AnimeGAN results proved to be difficult because the model had nine loss functions each with their specific weights. The paper unfortunately did not clearly state all the training parameters used and thus an unexpected amount of time was spent tuning these weights because the model results were highly sensitive to the relative weights between the different loss functions, particularly the relative weight between adversarial loss, grayscale adversarial style loss and grayscale edge promoting loss in the case of discriminator and adversarial loss, content loss and style loss in the case of the generator.

After fine tuning, the first iteration of AnimeGAN results can be found in Figure 4 (b) after training 40 epochs over 6500 images per epoch on the unpaired real world scene and anime scene dataset. Equal weights were applied to the adversarial loss and grayscale adversarial style loss to emphasize their equal importance to the discriminator. However, from the results shown, it is obvious that much of the original color was lost.

For the second iteration, we then increased the weight of the color loss in the generator to penalize large deviation in the generated image's YUV channels from that of the original and the grayscale edge promoting adversarial loss in the discriminator to capture more facial details. This result can be see in Figure 4 (c).

Unfortunately these results still performed poorly because clear facial features were not present and the results still look like a watercolor filter and blur tool was applied. To attempt to alleviate this, we switched the GAN loss type from the traditional sigmoid activation followed by binary cross entropy to least square used in LSGAN [14] where the loss is simply the $l2$ norm of the difference. This was another ambiguity that was not clarified in the paper nor the author's original source code as the author implemented multiple GAN loss functions but did not clearly state which one they ultimately used. Our motivation to switch to LSGAN was because it had more stable gradients, albeit also longer convergence times. This final result can be seen in Figure 4 (d) after 100 epochs of training and a smaller learning rate, where more of the facial features were capture.

In conclusion, we were not able to create realistic anime faces with AnimeGAN, despite our hyperparameter tuning experiments. One of the reason could simply be the lack of anime faces in the training data and the fact that the style used to draw anime faces is not the same style used to draw anime scenes, even though both are drawn by the same artist. Also the lack of anime face training data means the network did not actually know how to transform real human head shapes to anime head shapes, which is typically characterized by their pointed chins and slender jawlines. In fact, we are skeptical if AnimeGAN can ever capture this because its purpose was to understand style, not mappings of shapes. If time permitted, additional improvements such as continued training of the network on unpaired portraits and anime faces could potentially address this concern.

5.2. MUNIT

The original version of MUNIT was ran for 100,000 iterations with the default parameters on the train datasets. The end results, shown in Figure 1. were just starting to resemble anime characters for a specific subcategory. Some of the issues we noticed include the often cited deformed features and foreign artifacts on generated faces. The model outputted fairly diverse styles of generated images for each input image, which is what it promised to do. At the end of training, some facial features were still quite exaggerated, for example the eyes, which could possibly improve with more training iterations.

The altered version of MUNIT was ran for 50,000 iterations with the same parameters as the original MUNIT. The results are shown in figure 2.This method seems to require slightly more training as by iteration 20,000, facial features such as the eyes were still not clearly formed. However, the hair on the generated images does seem to be more smooth.

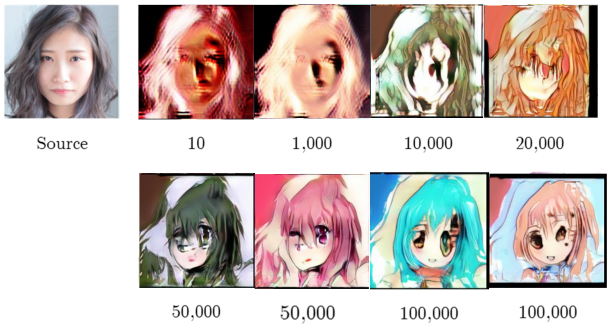

Figure 1. MUNIT results by iteration. Iteration refers to batch number, where each batch contains 1 pair of images. Two variations were shown for 50,000 and 100,000 iterations.

By 50,000 iteration, results from this hybrid approach looks close to results from original MUNIT with 100,000 iterations of training.

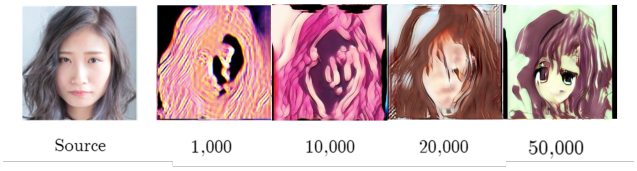

Figure 2. MUNIT with AniGAN decoder results by iteration. Iteration refers to batch number, where each batch contains 1 pair of images..

5.3. U-GAT-IT

Before committing time and computational resources full model training, we first performed experiments with the loss weights so see which combination performed the best qualitatively. The following ranges were tested, which were informed by the default hyperparameters:

Loss term	Weights tested
$\mathcal{L}_{adv}$	0.5, 1, 2
$\mathcal{L}_{cycle}$	5, 10, 20
$\mathcal{L}_{identity}$	5, 10, 20
$\mathcal{L}_{cam}$	1000, 2000

After testing the 54 models and their sample outputs, the best set of weights based on the generated samples was the tuple (2, 10, 20, 1000). This shows that for this particular task, the adversarial and identity losses are more important than the default. This could be due to the fact that in the reconstruction it is important to maintain some of the color statistics, like face and hair color.

Using this set of loss weights, the modified U-GAT-IT model was trained for 300,000 iterations, each of which is a gradient step performed on 1 pair of images from the training set. The progression of the generator can be seen in Figure 3. The generated images first start imitating the lower level features of anime images such as hair coloring and lines, and textures. Then it starts showing signs of more sophisticated patterns such as overall hairstyle, and facial features. What's interesting is that it goes through cycles in terms of color themes. Sometimes all predicted images will have a red tint and other times a green tint. This could be because with a batch size of 1 the generators sometime over fit if it receives too many similar Anime images for many iterations.

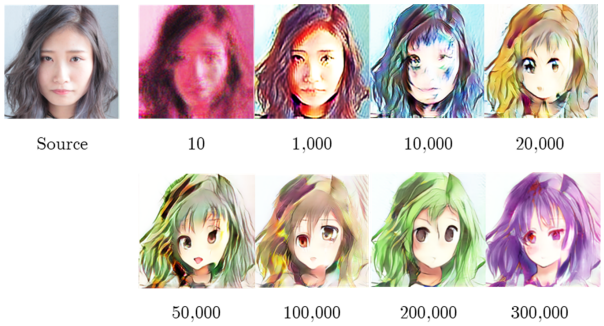

Figure 3. U-GAT-IT results by iteration. Iteration refers to batch number, where each batch contains 1 pair of images.

Overall the results are good compared to some of the other methods attempted. The facial structure of the generated images still lacks consistency (e.g. the eyes are uneven). With more training and perhaps better hardware and training procedure, the model will be able to achieve results similar to the original paper. The final model achieved a FID of **51.9** based on 5,568 generated images. This is slightly higher than the 42.8 reported for the original U-GAT-IT, 38.5 for AniGAN, and 50.0 for CycleGAN [13]. This is likely due to sub-optimal training and tuning as well as slightly different datasets being used: they used CelebA-HQ as the source domain instead of FFHQ.

5.4. Model Comparison

We then compiled the results from all 3 approaches and tested them on the same images to compare them qualitatively. These results can be seen in Figure 4.

As mentioned previously, AnimeGAN performed poorly compared to the other two methods due limitations in the training data and the lack of clarity in the appropriate loss weights and methodology. To effectively train a neural style transfer GAN network such as AnimeGAN on human portraits, we would first need a large dataset of anime faces drawn by the same artist to ensure style consistency. The Danbooru anime dataset used to train U-GAT-IT and MUNIT would unfortunately not help because it is a collection

of random anime styles from different artists.

U-GAT-IT and MUNIT are improvements on the CycleGAN architecture and performed better in image-to-image translation with unpaired datasets. Both methods start to show semblance of Anime facial features such as hairstyles, eye shapes, and facial structure. However, they still lack consistency in the generated images. Also, they oftentimes fail to maintain some of the features in the source image, such as hair color, eye color and skin color. That said, the U-GAT-IT results do seem to be the closest in terms of distribution to the actual Anime dataset. This could be a reflection of the layer normalization techniques or the loss function formulation. It also achieved the lowest FID score out of the 3 approaches, which can be seen in Table 1.

Model	AnimeGAN	U-GAT-IT	MUNIT
FID Score	135.91	51.96	58.01

Table 1. FID scores from the 3 approaches.

Another drawback that the U-GAT-IT and MUNIT approaches suffer from is that they are unable to distinguish female and male characters, more specifically, female and male features. For that reason, it tends to generate female-looking Anime characters even for male humans. This is likely due to two reasons: 1) the Danbooru dataset is heavily skewed towards female so the model is more likely to fit to a female portrait distribution, and 2) the model has a hard time capturing the semantic difference in gender as there is no gender information made available to the model.

6. Discussion

Overall, we were able to produce reasonable results with recently published methods, that achieve FID scores that are comparable with the state-of-the-art. The training and tuning process proved to be more difficult and computationally expensive than we were anticipating. We learned that the formulation of the loss function is very important in achieving good results, especially the inclusion of cycle consistency. There were also a few architectural considerations from a deep learning perspective that were essential to model performance. The top ones being the batch / layer normalization technique and the encoder-decoder design of the generators.

In terms of potential future work that the research community could focus on to make improvements in the image-to-image translation domain, we believe further research in loss function formulation could improve generation quality. In particular, we hope the various types of losses that are currently being used can be generalized and unified so that we only need to optimize one or two losses, thus minimizing the amount of tuning needed. Another area of advancement is in the model architecture. We see some usage of attention mechanisms and CAM in some recent research. We expect that this area could develop more substantially and lead to improvements in translation quality.

7. Work Division

Refer to Table 2 for an outline of each team member's contribution and details.

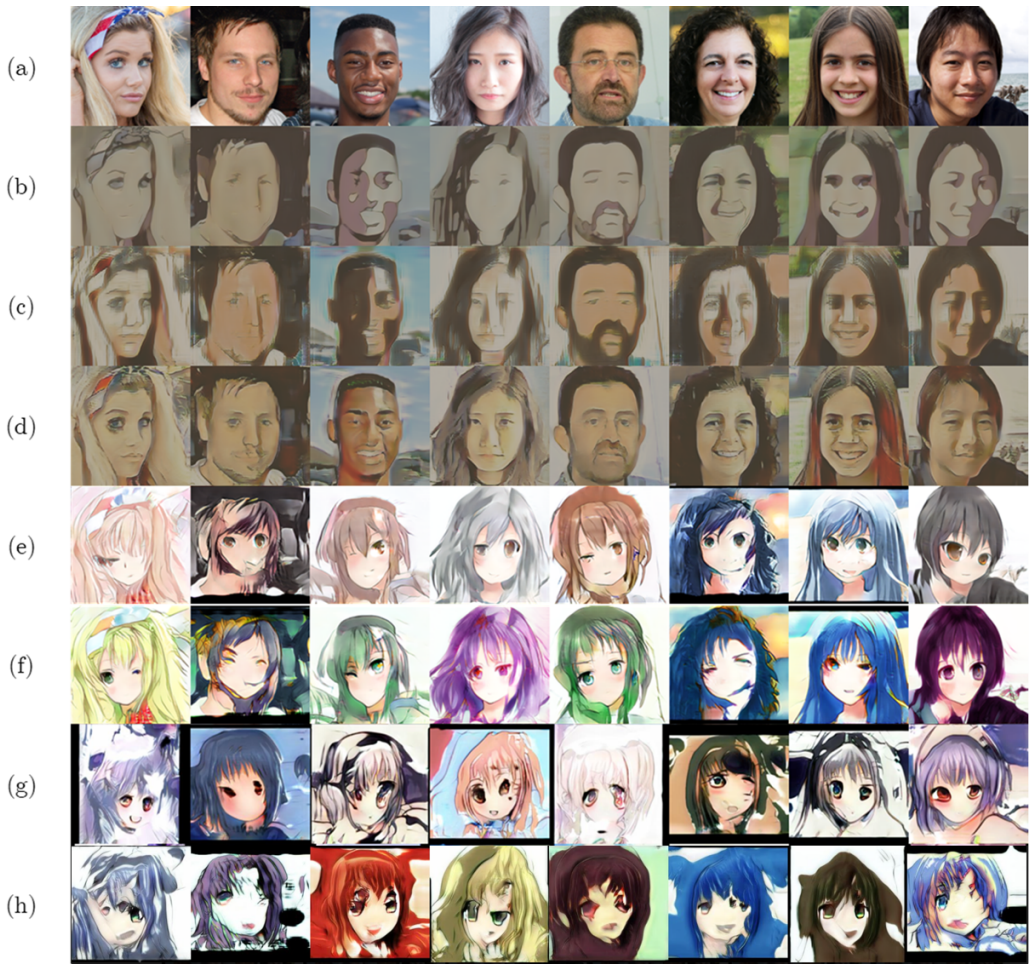

Figure 4. Comparison of translation results: (a) source image, (b) AnimeGAN after 40 epochs with equal weights in adversarial loss and grayscale adversarial style loss (c) AnimeGAN after improving upon (b) by increasing the weight of color reconstruction loss to better capture original color channels (d) AnimeGAN final results after 100 epochs and switching the GAN loss type from traditional sigmoid activation followed by binary cross entropy to least square as used in LSGAN, (e) U-GAT-IT at 150,000 iterations, (f) U-GAT-IT at 300,000 iterations, (g) MUNIT at 100,000 iterations, (h) hybrid MUNIT and AniGAN with AniGAN decoder at 50,000 iterations.

Student Name	Contributed Aspects	Details
Wesley Liao	Data Procurement and U-GAT-IT	Setup datasets in cloud storage and implemented U-GAT-IT in PyTorch. Experimented with loss parameters to improve results.
Gary Liu	AnimeGAN	Implemented AnimeGAN in PyTorch and conducted experiments to analyze various hypterparameter's effects on AnimeGAN convergence.
Nikki Hu	MUNIT and AniGAN	Trained AniGAN, MUNIT and a hybrid AniGAN-MUNIT architecture and analyzed the results.

Table 2. Contributions of team members.

References

[1] Gwern Branwen, Anonymous, and Danbooru Community. Danbooru2019 portraits: A large-scale anime head illustration dataset. https://www.gwern.net/Crops#danbooru2019-portraits, March 2019. 3

[2] Jie Chen, Gang Liu, and Xin Chen. *AnimeGAN: A Novel Lightweight GAN for Photo Animation*, pages 242–256. 05 2020. 2

[3] Jie Chen, Gang Liu, and Xin Chen. Animeganv2. https://github.com/TachibanaYoshino/AnimeGANv2, February 2021. 4

[4] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9465–9474, 2018. 1

[5] NVIDIA Corporation. Anigan. ://github.com/NVlabs/MUNIT/. 4

[6] Xun Huang, Ming-Yu Liu, Serge Belongie, and Ja Kautz. Multimodal unsupervised image-to-image translation, 2018. 3

[7] jis478. Anigan. https://github.com/jis478/AniGAN.git, April 2021. 4

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. 2

[9] Hyeonwoo Kang. U-gat-it — official pytorch implementation. https://github.com/znxlwm/UGATIT-pytorch, 2019. 4

[10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018. 3

[11] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwanghee Lee. U-GAT-IT: unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. *CoRR*, abs/1907.10830, 2019. 2

[12] Bryan Lee. Animeganv2 - pytorch. https://github.com/bryandlee/animegan2-pytorch, March 2021. 4

[13] Bing Li, Yuanlue Zhu, Yitong Wang, Chia-Wen Lin, Bernard Ghanem, and Linlin Shen. Anigan: Style-guided generative adversarial networks for unsupervised anime face generation, 2021. 3, 6

[14] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2017. 5

[15] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.1.1. 5

[16] Joost van Amersfoort, Wenzhe Shi, Alejandro Acosta, Francisco Massa, Johannes Totz, Zehan Wang, and Jose Caballero. Frame interpolation with multi-scale deep loss functions and generative adversarial networks, 2019. 1

[17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020. 2