Using Generative Adversarial Networks to Generate Mobile Game Artworks

Haoyang Zhu, Chuan Tian Zhang, Ron Li
{haoyang.zhu, ben.zhang, ron.li}@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Introduction

In the mobile gaming market, many games have a primary objective of collecting or gambling for special in-game characters and items. These games have become extremely prominent in the mobile game economy since they allow players to earn basic rewards for free. They can have up to thousands of in-game characters and items while new items are continuously released over time to maintain engagement and increase revenue. These in-game objects are created with the theme of the game in mind while also being different from other existing game characters. With these constraints in mind, it may be difficult for artists who produce art for the game to come up with new designs or draw interesting new characters consistently.

We propose to solve this problem with Generative Adversarial Networks (GANs), which have been used extensively for image synthesis. By building upon the existing art models and assets, it is possible to generate new content through the combination of these existing pieces using GANs. Similarly to how GANs can be used to generate faces based on images of people, we believe that they can be useful for generating fictional characters as well. This is especially true for images that share common characteristics. Fortunately, many free-to-play mobile games already have a large roster of characters and objects with detailed artworks that can be used for training.

We believe that there will be many real-world implications involved with this project. The first of which would be the possibility to apply this to an actual mobile game and help speed up their content delivery efficiency. This would in turn increase the content release cycles for these mobile games, making the game much more appealing compared to competitors. Another implication would be leveraging video game art synthesis to other areas in game development, potentially helping small developers generate art and asset ideas from other well-known games to give them inspiration on what to create. Finally, the results of this project can be used to fully automate content creation, making gaming experiences that are truly unique to the player while also greatly reducing development costs.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Contributions

The research problem that we are tackling is finding the best way to generate quality image assets for mobile games using various GAN models. During our preliminary investigation, we found several papers and articles for similar work done on the subject, however, the resulting generated assets are very blurry and hard to distinguish (Dennanni 2018). We hypothesize that this is likely due to the lack of adequate training data, in terms of both quality and quantity as well as underlying problems within the GAN model itself. Hence, using several new GAN architectures, we want to compare and contrast their results, speed, and complexity to better understand each model's effectiveness as well as its advantages and disadvantages.

Through our initial research, we found three different GAN models that could be leveraged to create an asset generation tool. These are Deep Convolutional GAN (DCGAN), ProGAN, and StyleGAN. We plan on gathering our dataset from websites and wikis hosting mobile game sprites. To ensure that our dataset is large enough, we plan on combining assets from multiple different games and use data augmentation techniques such as geometric transformations and introducing noise to ensure an adequate dataset size (Mikołajczyk and Grochowski 2018). To evaluate their effectiveness in creating art assets, we plan on using the Frèchet Inception distances (FID). At the same time, by using these GANs in our implementation, we can get a general sense of how complex each model is as well as how efficient they are based on the time it takes to train the model.

Originally we had anticipated that the different GAN models will each generate a unique but similar set of results, this hypothesis was confirmed by the results. After getting these results for each model, we were able to use the generated output to create performance evaluations and perform a series of aggregate analysis on these evaluations to draw some conclusions about the problem. We hope that by analyzing these evaluations, we can get a better sense of the general performance measures for each model and find pieces of each that can be used to augment our asset generation tool.

Our findings in this paper can be used to determine the viability of artificial asset synthesis for not just mobile video games, but also for any medium that places importance on digital artworks. It is possible that as research in this field progresses, this may cover all forms of digital art media.

Through our assessment of each GAN, we will also uncover the usefulness of Frèchet Inception distance to evaluate different GAN models on varying datasets. These insights will assist future work that may consider using the Frèchet Inception distance to evaluate machine-generated images and whether it would be a good choice compared to other alternatives.

Related Work

The concept of a Generative Adversarial network was created by Ian Goodfellow and his colleagues back in 2014 and is essentially an architecture where two neural networks work against one another. In this system, a neural network called the "generator" produces some output with the intent to fool a "discriminator" neural network that the output is legitimate. One particular application of a GAN is for image synthesis, having the "generator" create realistic and natural-looking images (Goodfellow et al. 2014). Over time, many varying implementations of GANs have been created with different takes on image synthesis.

In 2016, after a series of extensive model exploration, Alec Radford & Luke Metz identified a family of architecture that resulted in stable training across a range of datasets and allowed for training higher resolution and deeper generative models. This model, along with the architecture, became the basis for what is known as Deep Convolutional GANs. This approach builds upon three key changes to the traditional CNN (convolutional neural network) architecture. The first is the replacement of "deterministic spatial pooling functions ... with strided convolutions", which transposes the images from a minimized format to a larger one which then allows the network to learn its spatial downsampling for image compression. Next, is the "[elimination] of fully connected layers on top of the convolutional features" and directly connecting the highest convolutional features to the input and output. Lastly, DCGAN uses Batch Normalization to help "stabilize learning by normalizing the input to each unit to have zero mean and unit variance" (Radford, Metz, and Chintala 2015). Following this research, Lewis Horsley and Diego Perez-Lieban used this idea in their research for automatically creating sprites. In their paper, they explained the principles behind their idea for "a system that automatically generates new graphical assets". Leveraging DCGANs, they were able to generate sprites for three categories: Human-like sprites, faces, and creatures (Horsley and Perez-Lieban 2017). This work demonstrates the existence of some preliminary research into this field of asset generation, however, we believe that using other, more modern techniques, we can improve upon art generation with GANs.

Later on, a new type of architecture was designed to address image resolution issues. This architecture is called ProGAN and starts its training on lower resolution images, then progressively adds new higher-resolution layers to the network. In Nvidia's implementation, both the generator and the discriminator grow progressively in lock-step starting from 4x4 images and up to 1024x1024 images. Because most of the iterations are done at lower resolutions, results with comparable quality are obtained up to 2-6 times faster than with traditional GAN implementations. Moreover, the generation of smaller images in the early stages is more stable because of reduced class information and fewer modes (this is analyzed in a paper that the ProGAN paper references. Through progressively increasing resolution, the ProGAN architecture effectively utilizes the divide-and-conquer approach by "continuously [...] asking much simpler questions compared to the end goal of discovering a mapping from latent vectors to e.g. 1024^2 images." It turns out that this approach stabilizes the training sufficiently for the network to synthesize megapixel-scale images (Karras et al. 2017).

Finally, in late 2018, a team at NVIDIA proposed an alternative generator architecture for GANs called StyleGAN. Building on top of ProGAN which generates high-resolution images starting by low-resolution versions and progressively increasing the resolution, this team noted that many GAN architectures were lacking development in the generator neural network. Using AdaIN (Adaptive Instance Normalization) which allowed a "style" transfer between images in the generator, the new GAN architecture produced high-resolution images with excellent results as seen with the calculated Frèchet Inception distances (Karras, Laine, and Aila 2018). However, these results were from generating pictures of human faces and it is unsure whether StyleGAN can produce similar results with a different dataset.

In the few papers that we analyzed, we see a common trend: all of these papers aim to address the problem that GANs tend to become unstable when training with higher-resolution images. This may be analogous to the problem of finding local extrema with non-convex cost functions as we have seen in lectures. We are excited to explore this property of GANs in our project.

Methodology

For our research, we will utilize GANs to create and generate the art for mobile game characters. More specifically, we will experiment with the following GAN implementations: DCGAN, ProGAN, and StyleGAN2 (an improved version of StyleGAN). These architectures were chosen because they have been used in many experiments for image synthesis and have existing open-source implementations we can leverage.

Evaluation Method

When testing the effectiveness of each GAN implementation, we must use a sufficiently sized dataset for testing. This is mainly because neural networks rely on large datasets to train and also to prevent overfitting on a small dataset. As a guideline, NVIDIA's StyleGAN implementation trained on a dataset consisting of 70000 human faces as well as the LSUN datasets with millions of images, and both produced excellent results (Karras, Laine, and Aila 2018). In comparison, GANs have been trained on datasets fewer than 10000 images and even as low as under a thousand (Horsley and Perez-Lieban 2017). Although it seems that the quality of generated images with smaller datasets is visually worse than those generated from larger datasets, it is also possible the GAN architecture may be a key factor.

With the previous knowledge in mind, we need to acquire a dataset that is as large as possible to evaluate the effectiveness of each GAN architecture. Since mobile game artwork and sprites are not readily available as a packaged dataset due to private ownership of video game assets, we must create our dataset ourselves. Even though mobile game assets are technically privately owned, many games have publicly displayed assets on websites and wikis. We can ensure we get the most amount of data by creating a dataset from gacha-type mobile games that have earned the most profit in their years of operation, as well as those that have been around for a long time. This is because gacha-type games are designed with collecting in-game objects as a form of gameplay and will contain more in-game objects and art. Additionally, games that have been around longer and earn more profit will have more art and assets compared to those who were recently released and have lower revenue streams, meaning that we will be prioritizing older games when selecting the data.

To evaluate the results of training each GAN on our dataset, we have identified techniques such as Inception Score and Frèchet (FID). The inception score provides a quantitative evaluation of the quality of generated images through two criteria, having high overall diversity and each image being clear and well defined as opposed to being blurry (Barratt and Sharma 2018). However, it is more viable to use the FID instead since it is more consistent with human judgement and robust to noisy output (Lucic et al. 2017). It is also convenient that there are many open-source implementations on calculations FID that we will leverage for our evaluation.

Algorithms

Because the design of image GAN architecture is time-consuming and requires great knowledge in both computer vision and GANs. We have chosen three existing GAN architectures to evaluate. All of these architectures are popular for image generation and each has its unique capabilities.

DCGAN is the first GAN architecture that is explicitly designed for training stability. The paper identifies a set of restrictions on the architecture of Convolutional Neural Networks (CNNs) such that the training is stable. We chose this architecture because training stability is important in our time-limited research. Moreover, the DCGAN paper uses some popular CNN techniques such as strided convolutions, batch normalization, (Leaky) ReLU, and Tanh activation, which will help us better understand other techniques used in the next two papers (Radford, Metz, and Chintala 2015).

ProGAN is designed with speed and interpretability in mind. It starts its training on lower resolution images, then progressively adds new higher-resolution layers to the network. In Nvidia's implementation, both the generator and the discriminator grow progressively in lock-step starting from 4x4 images and up to 1024x1024 images. This will allow us to quickly obtain results during training, which is beneficial to understanding the architecture (Karras et al. 2017).

StyleGAN is one of the newest and most popular GAN implementations for image synthesis. Building on top of the ProGAN idea of steadily incrementing the resolution, it uses AdaIN to allow a "style" transfer between images in the generator. StyleGAN has worked extremely well on human faces and artificial images (e.g. cars and bedrooms). We are excited to see how it performs on the dataset of game characters. For our particular study, we will be using StyleGAN2. This is an improved version of StyleGAN which yields improved FID scores (Karras, Laine, and Aila 2018; Karras et al. 2019a).

Results

We trained all three GAN architectures on up to 6 datasets (with the exception of ProGAN, where we trained on fewer datasets due to resource constraints). Three datasets contain character artworks and two datasets contain in-game sprite assets. The datasets are as follows: PAD (Puzzles and Dragons Character Artworks 6300 Images), BF (Brave Frontier Character Artworks with 2400 Images), FE (Fire Emblem Character Artworks with 576 Images), FE portraits (A 500 image subset of FE with consistent character poses), GBF (Granblue Fantasy Character Sprites with 1200 Images), FGO (Fate Go Character Sprites with 800 images)

DCGAN

Early iterations of the DCGAN training showed promising FID scores as shown in Figure 1, however, prolonged DCGAN training seems to be unstable with little to no improvement (Figures 5 and 9). As the resolution of images used for DCGAN increased, so too did this instability, for this reason, we decided to perform training on only 64x64 images. We make use of the implementation found in the Pytorch examples and obtained very optimistic FID (Frechet Inception Distance) scores (which we will outline in the next few sections). (Chintala et al. 2016). The results are surprisingly good and achieved in a relatively short amount of time compared to the other GAN techniques, especially when our training was done on an older GTX 1660 GPU. The dataset is mirror augmented to double the number of images available.

PAD Dataset Figure 1 shows some sample output from DCGAN after training on over 200 thousand images from the PAD dataset. Figure 2 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. Lowest FID achieved is 54.

BF Dataset Figure 3 shows some sample output from DCGAN after training on over 100 thousand images from the BF dataset. Figure 4 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. Lowest FID achieved is 72.

FE All Dataset Figure 5 shows some sample output from DCGAN after training on over 170 thousand images from the FE All dataset. Figure 6 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. Lowest FID achieved is 87.

FE Portraits Dataset Figure 7 shows some sample output from DCGAN after training on over 170 thousand images

Figure 1: Example output of DCGAN after training on 200 thousand images from the PAD dataset

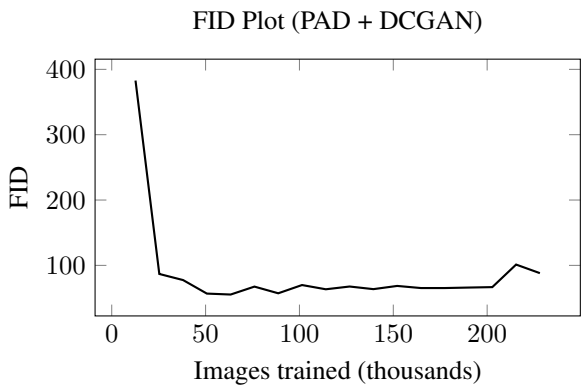


Figure 2: Frèchet Inception Distance over the number of images seen by the network (DCGAN trained on the pad dataset)

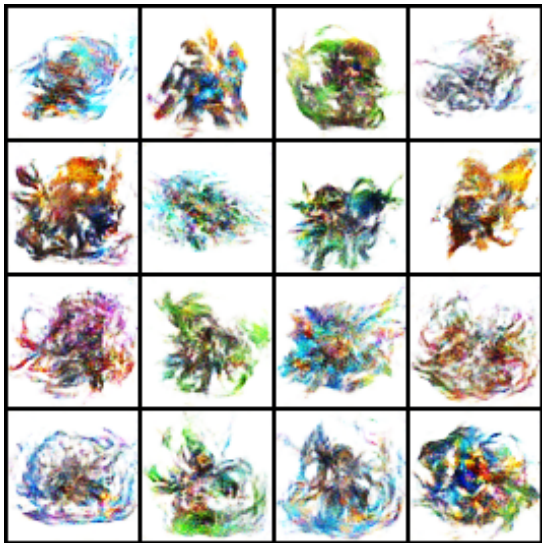


Figure 3: Example output of DCGAN after training on 100 thousand images from the BF dataset

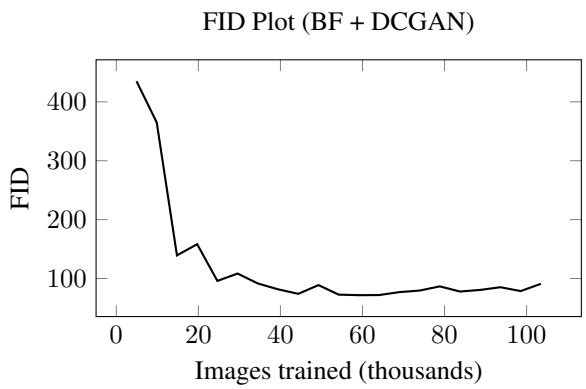


Figure 4: Frèchet Inception Distance over the number of images seen by the network (DCGAN trained on the BF dataset)

from the FE dataset. Figure 8 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. Lowest FID achieved is 87.

GBF Dataset Figure 9 shows some sample output from DCGAN after training on over 70 thousand images from the GBF dataset. Figure 10 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. Lowest FID achieved is 74.

FGO Dataset Figure 11 shows some sample output from DCGAN after training on over 45 thousand images from the FGO dataset. Figure 12 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. Lowest FID achieved is 114.

In general, the DCGAN training produced some of the best results at the earlier epochs. The worst results were usually at the very beginning and the very end, at which point the training grew too unstable to continue. This instability can be seen most clearly in the training results of the FE portraits dataset where the FID scores fluctuated significantly between every dozen epochs (See Figure 8). Looking at the actual results, we can see that the trained models tended towards a heavily pixelated and color skewed version that eventually went back to normal (See Figure 13).

Initially, we hypothesized that DCGANs will produce stable training results. However, it would appear that the results indicate otherwise. Fortunately, DCGAN does have its benefits, mainly in that it can produce images with relatively low FID scores in the first few dozen epochs. This leads us to believe that DCGAN would be a good candidate for getting decent results very quickly. However, due to the instability described earlier, it would be much more difficult to refine the results and improve upon it with further training, as the FID scores rarely improve beyond the initial 50 epochs.

Figure 5: Example output of DCGAN after training on 170 thousand images from the FE dataset

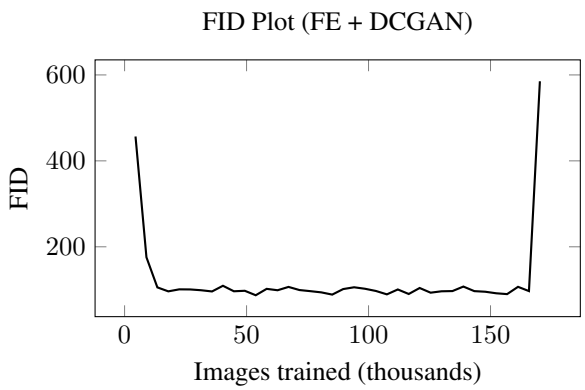


Figure 6: Frèchet Inception Distance over the number of images seen by the network (DCGAN trained on the FE All dataset)

From the results, we can identify a general pattern among all datasets where the FID scores improve very quickly within the first few dozen epochs, then make barely any improvement at all after a certain point. We hypothesize that the reasoning behind why the DCGAN results follow this pattern can be attributed to several factors to which we configured the training. First of which is the dimensions of the training images. As the resolution increases, so too does the requirement for better hardware with which to train the data. For DCGAN, this training was done on a Nvidia GTX1660 GPU, which could only handle 64 by 64 pixel images. As a result, the lowered resolution meant that it is much more likely for the trained images to look similar to the actual images, potentially inflating the FID scores. Another factor contributing to the DCGAN results can be the size of the datasets. Comparing the FID score results of the different datasets, the PAD dataset produced some of the best scores.

Figure 7: Example output of DCGAN after training on 170 thousand images from the FE Portraits dataset

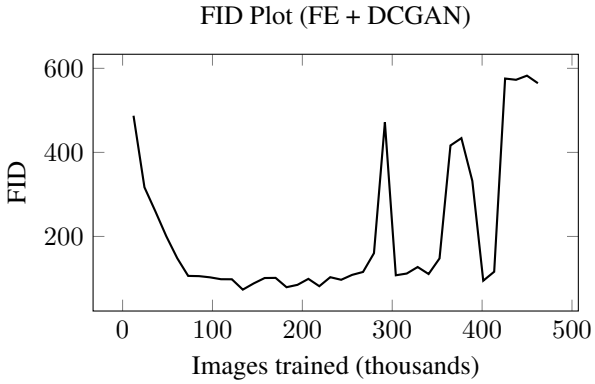


Figure 8: Frèchet Inception Distance over the number of images seen by the network (DCGAN trained on the FE Portraits dataset)

One reason for this could be the number of sample images available from which DCGAN could train. The PAD dataset contained around 6,000 images, which, in comparison to the BF dataset or FE dataset that contained only around 2,000 images, meant that there was roughly triple the amount of training variety available. This reasoning also aligns with our understanding of neural networks from course content where having large datasets produced better results.

ProGAN

The stability of ProGAN is well-demonstrated during our testing. In all of the training runs, the network is able to converge with a decent FID score. The FID curves of all training runs resemble an inverted step function, where the FID score alternates between leveling off and sharply decreasing (Figures 15, 17 and 20).

The best results for ProGAN are obtained from training

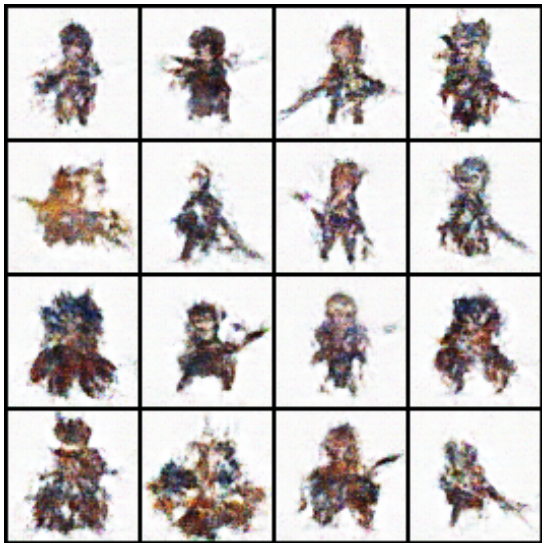


Figure 9: Example output of DCGAN after training on 70 thousand images from the GBF dataset

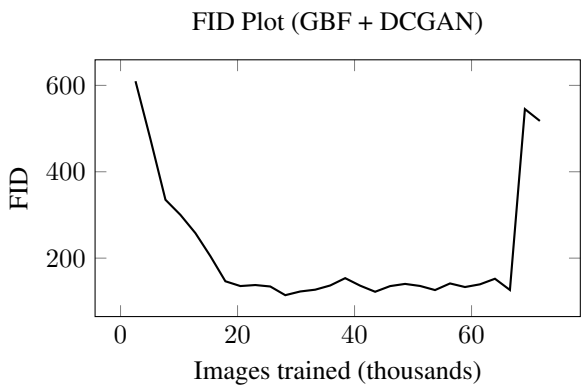


Figure 10: Frèchet Inception Distance over the number of images seen by the network (DCGAN trained on the GBF dataset)

Figure 11: Example output of DCGAN after training on 45 thousand images from the FGO dataset

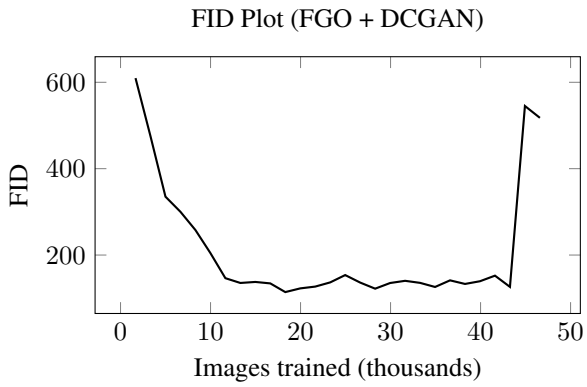


Figure 12: Frèchet Inception Distance over the number of images seen by the network (DCGAN trained on the FGO dataset)

on the FE Portraits dataset, where high-level details such as body shape and clothing contours are well-captured (Figure 16). The performance of ProGAN on this dataset is confirmed by its lowest FID score (79, Figure 17) across all 3 implementations that we experimented with. However, during our training, ProGAN was not able to capture finer details such as facial expressions or more granular body parts such as fingers. In the ProGAN paper, the authors are able to achieve high-quality image generation for bedrooms, which is of higher granularity than our training examples. This may suggest that it requires higher-resolution training data and longer training time.

On sprite datasets such as GBF, however, ProGAN generated relatively undesirable results (Figure 19). This may be because of the heterogeneity of the input data. On portrait datasets, every image had the same features in the same places (i.e. every image has a head above the body, and hands and legs around the body), whereas on sprite datasets, there is a lot more room for artistic variations. Moreover, the various weapons that the characters hold may also make it more difficult for ProGAN to find similarities.

BF Dataset Our BF dataset contains 2400 512px-by-512px images. We mirror augment it to double the number of images available. ProGAN begins training on a downsized 4x4 version of the dataset and progressively increases the image resolution until it reaches 512x512. By the end of our 32-hour training, we are able to generate 128x128 images with the network.

Figure 14 shows some example output from ProGAN after training on 6.72 million images from the BF dataset. Figure 15 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. The FID appears to be flat most of the time with rapid decreases periodically. This may be related to what we learned in class about get-

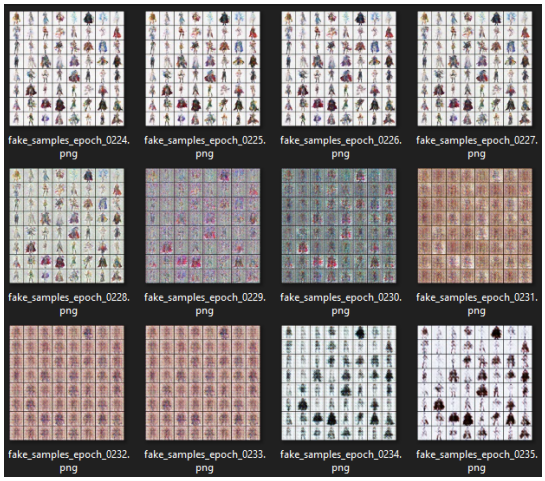


Figure 13: Training instability in DCGAN results

ting stuck in local optima of the search space. It may also be because ProGAN periodically doubles the input/output resolutions (we are using the default setting of doubling the resolution and fading in higher resolution images every 1.2M images). The lowest FID score achieved during our training is 99.2.

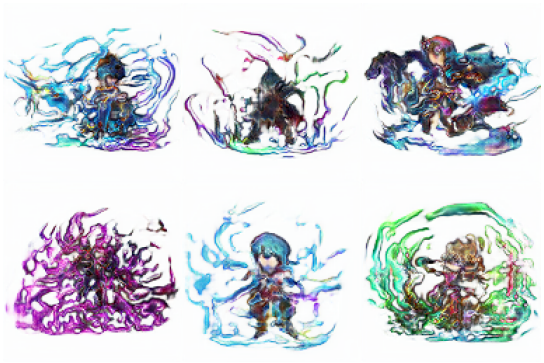


Figure 14: Example output of ProGAN after training on 6.72 million images from the BF dataset

FE Portraits Dataset ProGAN performed the best on the FE Portraits dataset. This may be due to the homogeneity of this dataset (every image contains a character that is in roughly the same pose). Figure 16 shows a few example output of ProGAN after training with 12 million images. Figure 17 shows the FID curve. Similar to the BF Dataset, the FID decreases in "steps" and eventually converges. The lowest FID is 79.3, and is obtained near the end of the training.

GBF Dataset ProGAN performed well on the GBF Dataset. As we can see from Figure 19, the characters are mostly clear and details such as thin swords are pronounced. However, there is a lot of unnecessary curved lines in the output. This may be because the input data has lots of variations (the characters may be in different poses and holding different weapons). However, surprisingly, the eyes of the

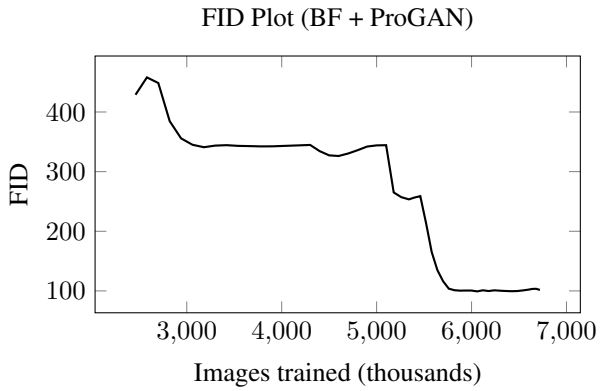


Figure 15: Frèchet Inception Distance over the number of images seen by the network (ProGAN trained on the BF dataset)

Figure 16: Example output of ProGAN after training on 12 million images from the FE Portraits dataset

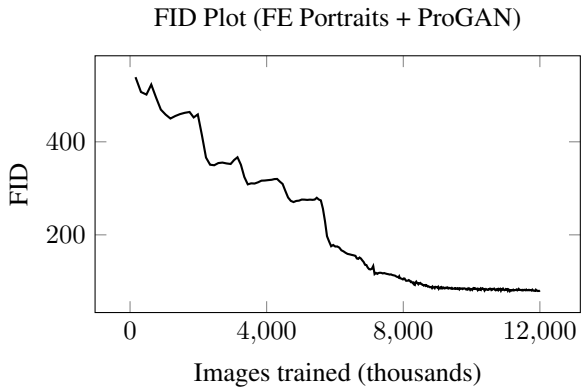


Figure 17: Frèchet Inception Distance over the number of images seen by the network (ProGAN trained on the FE Portraits dataset)

characters are well captured in the output. This may be because there is a relatively consistent way to draw eyes in our dataset. The FID curve shown in Figure 20 is similar to other FID curves for ProGAN. It follows a step shape and eventually converges.

(a) Generated Images (b) Training Images

Figure 18: ProGAN-generated images compared with the training images in the FE Portraits dataset.

Figure 19: Example output of ProGAN after training on 10 million images from the GBF dataset

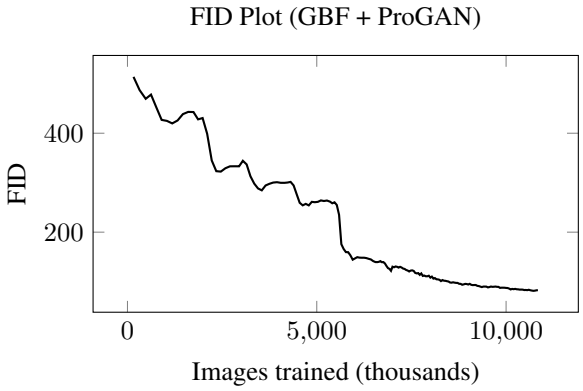


Figure 20: Frèchet Inception Distance over the number of images seen by the network (ProGAN trained on the GBF dataset)

StyleGAN2

All datasets are trained on the StyleGAN2 Tensorflow implementation by NVIDIA using config-f, which is the configuration that was originally used to train the Flickr-Faces-HQ Dataset. The training done in the paper is on a larger dataset at 1024x1024 resolution which requires many weeks of training on 1 Nvidia Tesla V100 GPU. (Karras et al. 2019b). We only have access to a single Nvidia RTX 2080Ti GPU so we decided to train at a lower resolution (512x512). One exception is that the GBF dataset which was trained at 256x256 because the training images were smaller. Each dataset was trained until the FID score remained constant for 200k images. In addition, each dataset is mirror augmented to double the number of images available for training.

PAD Dataset Figure 21 shows some sample output from StyleGAN2 after training on over 3 million images from the PAD dataset. Figure 22 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. The lowest FID score achieved is 97.

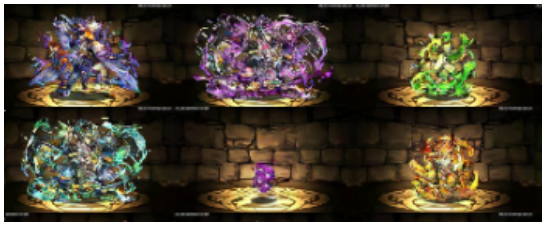


Figure 21: Example output of StyleGAN2 after training on 2 million images from the PAD dataset

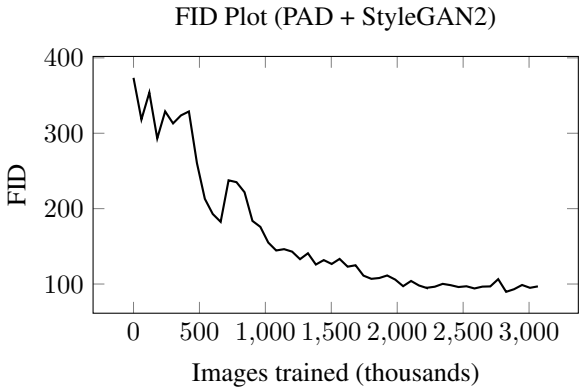


Figure 22: Frèchet Inception Distance over the number of images seen by the network (StyleGAN2 trained on the PAD dataset)

BF Dataset Figure 23 shows some sample output from StyleGAN2 after training on over 3 million images from the BF dataset. Figure 24 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. The lowest FID score achieved was 50.

FE Dataset Figure 25 shows some sample output from StyleGAN2 after training on over 2.3 million images from the FE dataset. Figure 26 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. The lowest FID score achieved was 120.

FE Dataset Subset: FE Portraits Figure 27 shows some sample output from StyleGAN2 after training on over 0.8 million images from the FE portraits dataset. Figure 28 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. The lowest FID score achieved was 180.

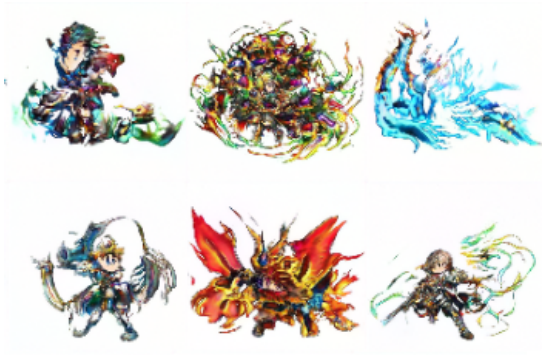


Figure 23: Example output of StyleGAN2 after training on 3 million images from the BF dataset

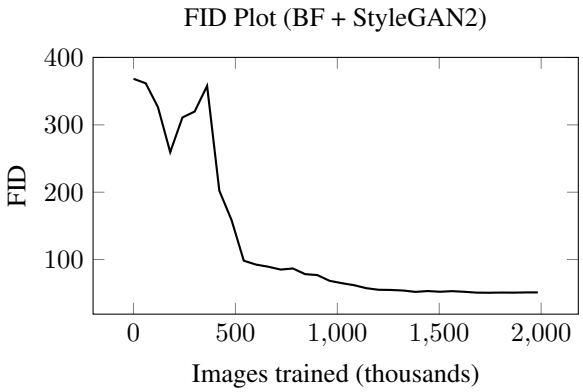


Figure 24: Frèchet Inception Distance over the number of images seen by the network (StyleGAN2 trained on the BF dataset)

Figure 25: Example output of StyleGAN2 after training on 2.3 million images from the FE dataset

GBF Dataset Figure 29 shows some sample output from StyleGAN2 after training on over 2 million images from the GBF dataset. Figure 30 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. The lowest FID score achieved was 50.

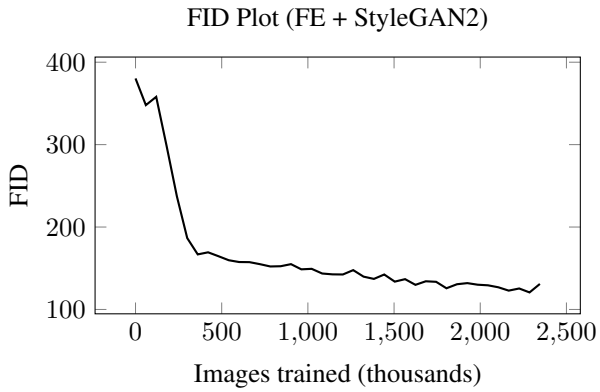


Figure 26: Frèchet Inception Distance over the number of images seen by the network (StyleGAN2 trained on the FE dataset)

Figure 27: Example output of StyleGAN2 after training on 0.8 million images from the FE Portrait dataset

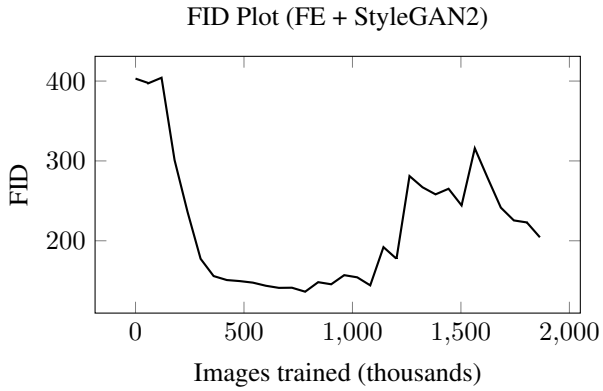


Figure 28: Frèchet Inception Distance over the number of images seen by the network (StyleGAN2 trained on the FE Portraits dataset)

FGO Dataset Figure 31 shows some sample output from StyleGAN2 after training on over 0.7 million images from

Figure 29: Example output of StyleGAN2 after training on 2 million images from the GBF dataset

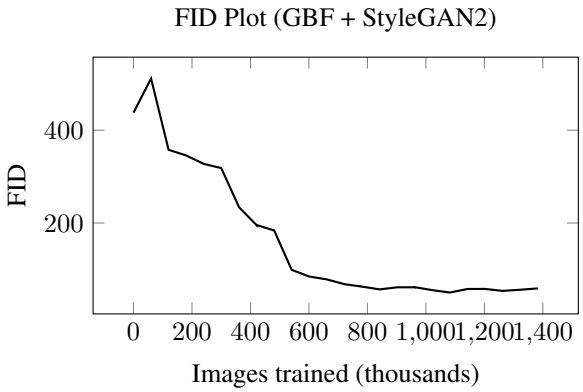


Figure 30: Frèchet Inception Distance over the number of images seen by the network (StyleGAN2 trained on the GBF dataset)

the FGO dataset. Figure 32 is a plot of the Frèchet Inception Distance (FID) over the number of images seen by the network. The lowest FID score achieved was 139.

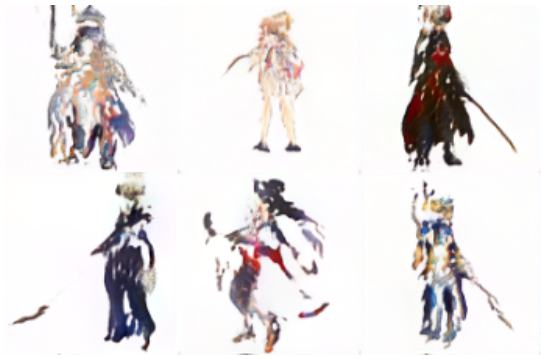


Figure 31: Example output of StyleGAN2 after training on 0.7 million images from the FGO dataset

StyleGAN2 was one of the most resource-intensive GAN architecture out of the three tested GANs because the research done to create it is more recent and had access to

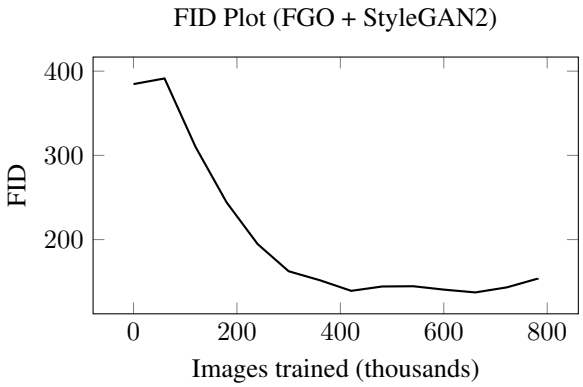


Figure 32: Frèchet Inception Distance over the number of images seen by the network (StyleGAN2 trained on the FGO dataset)

the latest hardware. Although the training was done on a more powerful GPU, it took significantly longer with millions of images trained for the FID scores to stop improving. The result is that better FID scores were achieved for most of the datasets and comparatively better-looking image results. Since StyleGAN2 leverages the progressive resolution increases during training similarly to ProGAN, it was not only able to produce high-resolution images but images with varying amounts of detail. Training curves for StyleGAN2 typically had a sharp drop in FID followed by a slow decrease. Although it is possible that training even further or increasing the training rate in the configuration may have allowed better FID scores, due to the time constraints of training multiple datasets we were unable to experiment with hyperparameters.

The results for the BF and GBF datasets had the lowest FID scores compared to the other datasets. Interestingly, these are the two datasets where characters lack a humanoid appearance since images represented a downscaled video-game-esque representation of a character. We hypothesize that this may be because StyleGAN2 may not be as effective on images with complex body poses such as the FE and FGO dataset. This is reinforced by the fact that the PAD dataset also had a lower FID score than FE and FGO. However, this could also be a shortcoming of FID as a metric to evaluate the effectiveness of StyleGAN2. Although PAD produced a lower FID score, the images produced visually did not represent the original dataset well since most outputs were simply a jumble of colours. We can see a detailed example in Figure 33.

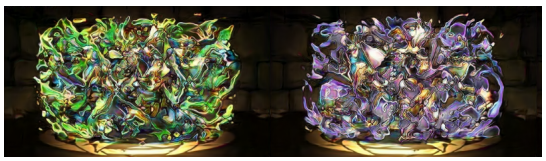
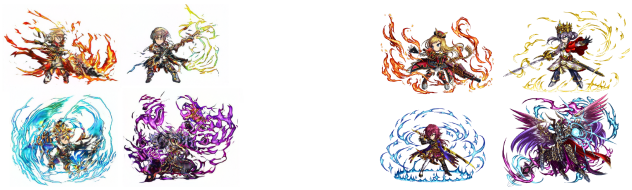


Figure 33: Examples of common blob-like images in the PAD output



(a) Generated Images (b) Training Images

Figure 34: StyleGAN2-generated images compared to the training images in the BF dataset.

From a visual perspective, StyleGAN2 produced many convincing images for the FE, GBF, and BF datasets. These generated images are visually similar to the style and theme of the original dataset when observing from a distance as seen in Figure 34. Unfortunately, once zoomed in we see that StyleGAN2 still has many issues generating the finer details since they will be shown as many small blobs as seen in Figure 35. The stochastic variation demonstrated in the original StyleGAN paper (Karras, Laine, and Aila 2018) may be the cause of the blobs here because StyleGAN was trained on 1024x1024 resolution human face images which can be much simpler than the art of a full character.

Figure 35: Example of a close-up image from the BF StyleGAN2 output

Comparison between Architectures

In summary, all of the GAN implementations have their pros and cons. DCGAN trains orders of magnitude faster than ProGAN and StyleGAN2, even on lower-end hardware. However, it is prone to instabilities, which constrains its ability to generate high-resolution output. ProGAN addresses the stability issue by training on progressively higher resolution images, but its training time becomes slower as a result. StyleGAN2 improves upon the ProGAN implementation to vastly reduce the time needed for FID convergence. Moreover, StyleGAN2 provides excellent visual variety in images that reflect the diversity of the dataset. It also introduces a way to map a set of real numbers to generated images, which enables interpolation between various attributes such as hair color, body shape, etc. One thing that all 3 GANs didn't capture during our testing is the fine details in the input images, such as fingers and facial features. This may be because our training sets have too many macroscopic features such as arms, legs, and the body, which hinders the networks' ability to focus on the smaller details.

Conclusion

In this paper, we evaluated 3 novel GAN architectures for generating portraits and sprites for humanoid characters found in video games. These GAN architectures improve upon existing research for video game asset generation, where the results are often blurry and hard to distinguish.

The methodology we used to tackle the problem was to compare the effectiveness of three GAN implementations, DCGAN, ProGAN, and StyleGAN. To help get a better variety of results, we wanted to train the three implementations on several unique datasets from mobile games of the same genre but with different art styles. We also wanted to make the distinction between differences in game art and sprite assets, since they look different fundamentally and should thereby be evaluated differently. More specifically, these datasets include artworks from Puzzles and Dragons, Brave Frontier, and Fire Emblem as well as sprite assets from Granblue Fantasy and Fate Go. After training the GAN models on the different datasets, we plan on using the Frechet Inception distance to evaluate the effectiveness of each model. By doing so, we will be able to gauge the performance of the different GAN models against each other using an objective metric rather than a subjective view.

While training on the three GAN models, we also kept track of the FID scores for the training results. Afterwhich, we plotted the FID scores against images trained to chart the results. For DCGANs, the training results showed very promising FID scores for the initial epochs, getting to as low as 55 for the PAG dataset. Unfortunately after reaching a low FID score averaging 70 at around the 50th epoch or 20-30k images, subsequent training yielded minimal to no improvements. In terms of ProGAN, training was much more stable, with FID scores converging on relatively low values of around 100. However, training is incredibly slow and the score is only achieved after many hours of training (upwards of 8 hours over around 5 million images). As for the last model, StyleGAN 2, it was able to achieve the lowest FID score of round 50 on the BF and GBF datasets. In addition, it also took a significant amount of training time, but the FID score converged after a fewer number of images trained when compared to ProGAN. Unfortunately, StyleGAN 2 experienced issues when training on small datasets or those with many small details where the generated images can be visually inaccurate.

Looking at the results as a whole, it can be said that DCGAN is most effective at generating small assets in a very short amount of time, ProGAN is great for stable image generation that can have greater detail at the cost of longer training times, and StyleGAN 2 can generate stable, high-resolution images at a decent speed with a tradeoff

for small, detailed datasets. In general, between the 3 GAN techniques, there are benefits and drawbacks to each, hence, for the mobile gaming industry, choosing the right technique depends significantly on the circumstances and the requirements needed.

Future Work

Given the time and resource constraints, there were many experiments we wished to perform that we were unable to. If we were able to continue with the project, we would try different configurations of hyperparameters on each GAN architecture and train for much longer to ensure that we truly achieved the best possible results on each dataset. More importantly, we would want to evaluate the results with more than the FID metric. As we found in the results, the FID metric may not be the best in determining if the training has finally reached the best possible result. Therefore for future work, we would consider using a multitude of metrics rather than depending on one.

Because our choice of datasets may have also impacted our results in this study, we would also want to consider training on different types of mobile game artwork. In this paper, we focused on character-based art in gacha-type games due to the availability of data. However, many other mobile games have vastly different art assets (such as weapons and items) that could potentially be used as an experiment. Possible methods of acquiring these datasets would be through data mining or perhaps direct contact with the developers. Being able to train on a wider variety of game art assets would be able to provide stronger conclusive evidence on the viability of GANs for image synthesis.

References

[Barratt and Sharma 2018] Barratt, S., and Sharma, R. 2018. A note on the inception score.

[Chintala et al. 2016] Chintala, S.; Lerer, A.; McCann, B.; and Varma, R. 2016. Deep convolution generative adversarial networks. https://github.com/pytorch/examples/tree/master/dcgan.

[Dennanni 2018] Dennanni, A. 2018. Creating pokémon with deep learning.

[Goodfellow et al. 2014] Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks.

[Horsley and Perez-Lieban 2017] Horsley, L., and Perez-Lieban, D. 2017. Building an automatic sprite generator with deep convolutional generative adversarial networks.

[Karras et al. 2017] Karras, T.; Aila, T.; Laine, S.; and Lehtinen, J. 2017. Progressive growing of gans for improved quality, stability, and variation.

[Karras et al. 2019a] Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2019a. Analyzing and improving the image quality of stylegan.

[Karras et al. 2019b] Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2019b. Stylegan2 — official tensorflow implementation. https://github.com/NVlabs/stylegan2.

[Karras, Laine, and Aila 2018] Karras, T.; Laine, S.; and Aila, T. 2018. A style-based generator architecture for generative adversarial networks.

[Lucic et al. 2017] Lucic, M.; Kurach, K.; Michalski, M.; Gelly, S.; and Bousquet, O. 2017. Are gans created equal? a large-scale study.

[Mikołajczyk and Grochowski 2018] Mikołajczyk, A., and Grochowski, M. 2018. Data augmentation for improving deep learning in image classification problem.

[Radford, Metz, and Chintala 2015] Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks.