

WHITEPAPER



Revealing the Secrets of Creating New Game Characters Without Using Paintbrush

Author: Dr. Monika Singh, Sourav Mazumdar

Contents

| | |
|--|----|
| Introduction | 02 |
| Types of GAN | 03 |
| Architectural details of StyleGANs | 06 |
| Business objectives and possible solutions | 08 |
| Experimental setup | 09 |
| Experiments results | 10 |
| Conclusion | 13 |
| References | 14 |

Introduction

In the gaming industry, the studios are responsible for the design and development of any game. A game designer will be responsible for providing multiple design ideas for various In-game components like characters, maps, scenes, weapons, etc. To develop a single character, a designer has to factor in multiple attributes like face morphology, gender, skin tone, clothing accessories, expressions, etc leading to a long and tedious development cycle. To minimize this complexity, we aim to identify tools and techniques, which can combine the automation power of machines to generate designs based on certain guard rails defined by the designers. This approach will be a path towards machine creativity with human supervision. From a business point of view, there will be more design options for the studios to select within a short span of time leading to huge cost savings.

Our solution utilizes advantage of advance deep learning models like GANs which have been proven to work extremely well in generative tasks (generating new data instances) such as image generation, image-to-image translation, voice synthesis, text-to-image translation, etc.

In this whitepaper, we explore the effectiveness of GANs for a specific use case in the gaming industry. The objective of using GAN in this use case is to create new Mortal Kombat (MK) game characters through style transfer on new or synthesized images and conditional GANs to generate only the characters of interest.

GAN & it's clever way of training a generative model for image generation!

"One of the remarkable applications of neural networks is the Generative Adversarial Network(GAN), which can be used for generative tasks. Introduced in 2014, GAN's are used in multiple application areas, like new images generation, image segmentation, voice synthesis, text summarization, etc."

GAN has two competing neural networks, namely a Generator (G) and a Discriminator (D). In the case of image generation, the goal of the generator is to generate images that are indistinguishable

(or the fake images) from the training images (or the real images). The goal of the discriminator is to classify between the fake and real images. The training process aims towards making the generator fool the discriminator, hence, to get the generated fake images that are as realistic as the real ones.

Following are the highlights of the GANs solution framework to create new Mortal Kombat Game characters without using a paintbrush:

- Showing detailed analysis and the effectiveness of GANs to generate new MK characters in terms of image quality produced (subjective evaluation) and FID distances (objective evaluation).
- Results of style mixing using MK characters. The style mixing performed using the trained model.
- Experimental evaluation using Mortal Kombat dataset (custom dataset having 25,000) images. The training time is captured to understand the computation resources required to achieve the desired performance.

Types of GAN and their role in creating Mortal Kombat realistic characters

GANs are an advanced and rapidly changing field backed by unsupervised machine learning methodology. In order to use GAN effectively to create Mortal Kombat realistic characters, it is vital to comprehend its architecture and different types in use to get near-perfect results. In this section, we will discuss the types of GAN and architectural details of StyleGans in the latter stage.

Types of GAN

GAN (or vanilla GAN) [Goodfellow et al. 2014]-GAN belongs to a class of methods used for learning generative models based on game theory. There are two competing neural networks, Generator (G) and Discriminator (D). The goal of GAN is to train a generator network to produce sample distribution, which mimics the distribution of the training data. The training signal for G provided by the discriminator D, that is trained to classify the real and fake images. The following is the cost function of GAN.

Cost function:

$$\min_G \max_D \left(\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \right).$$

The min/max cost function aims to train the D to minimize the probability of the data generated by G (fake data) and maximize the probability of the training data (real data). Both G and D are trained in alternation by Stochastic Gradient Descent (SGD) approach.

THE PROBLEM

Though GAN provides a powerful mechanism to train a generator to mimic the training distribution using training signals from discriminator, certain limitations make it less effective in the case of image generation.

LIMITATIONS

- Instability in training-Mode collapse is an inherent problem in the GAN generator, which occurs when the generator doesn't learn any useful representation which is indistinguishable from the real images, thus making the discriminator always win.
- Capable of generating relatively smaller images, such as 64*64 pixels.

THE SOLUTION

These limitations are overcome by using a more sophisticated model, known as Progressive Growing GAN (ProGAN), specifically designed to minimize the limitations of vanilla GANs.

Progressive Growing GAN(ProGAN) - ProGANs (Karras et al., 2017) are capable of generating high-quality photorealistic images, starting with generating very small images of resolution 4*4, growing progressively into generating images of 8*8,...1024*1024, until the desired output size, is generated. The training procedure includes cycles of fine-tuning and fading-in, which means that there are periods of fine-tuning a model with a generator output, followed by a period of fading in new layers in both G and D.

Style GANs - StyleGANs (Karras et al., 2019) are based on ProGANs with minimal changes in the architectural design to equip it to demarcate and control high-level features like pose, face shape in a human face, and low-level features like freckles, pigmentation, skin pores, and hair. The synthesis of new images controlled by the inclusion of high-level and low-level features. And it is executed by a style-based generator in styleGAN. In a style-based generator, the input to each level is modified separately. Thus, there is a better control over the features expressed at that level.

There are various changes incorporated in styleGAN generator architecture to synthesize photorealistic images. These are bilinear up-sampling, mapping network, Adaptive Instance Normalization, removal of latent point input, the addition of noise, and mixing regularization control. The intent of using StyleGAN here is to separate image content and style content from the image.

(Note: For generating MK characters, we have used styleGAN architecture, and the architectural details are provided in the next section.)

Conditional GANs - If we need to generate new Mortal Kombat characters based on the description provided by end-users. In vanilla GANs, we don't have control over the types of data generated. The purpose of using conditional GANs [Mirza & Osindero, 2014] is to control the images generated by the generator based on the conditional information given to the generator. Providing the label information (face mask, eye mask, gender, hat, etc.) to the generator helps in restricting the generator to synthesize the kind of images the end-user wants, i.e. "content creation based on the description".

Cost function:

$$\min_G \max_D \left(\mathbb{E}_{\mathbf{x}, \mathbf{y} \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})} [\log D(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{y} \sim p_y, \mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}, \mathbf{y}), \mathbf{y}))] \right).$$

This conditional information is supplied "prior" to the generator. In other words, we are giving an arbitrary condition 'y' to the GAN network, which can restrict G in generating the output and the discriminator in receiving the input. The following figure depicts the cGAN inputs and an output.

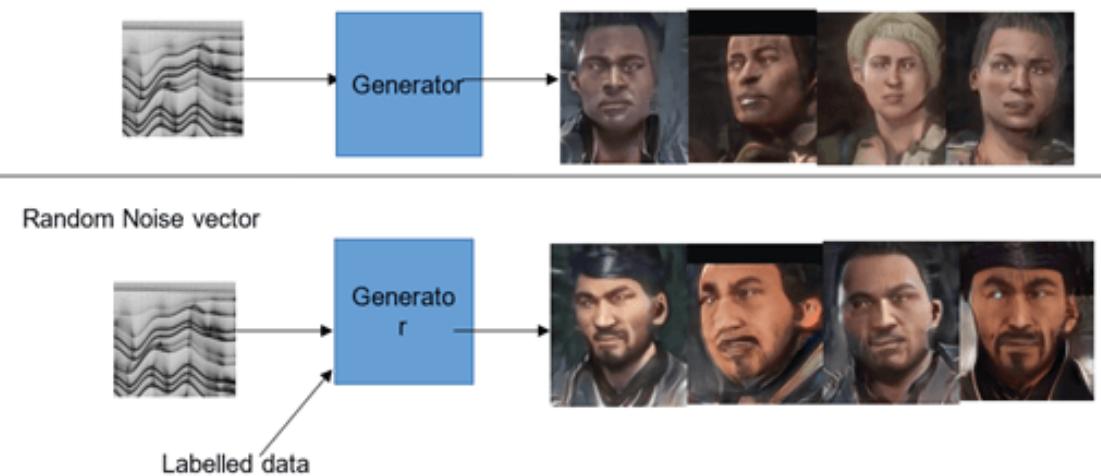


Fig 1. A- Random generation of MK characters using Random Noise vector as input, B-Controlled generation of MK characters using Random Noise vector and labeled data as input (Condition-Create MK characters with features as Male, Beard, Moustache)

Architectural details of StyleGANs

Following are the architectural changes in styleGANs generator:

- The discriminator D of a styleGAN is similar to the baseline progressive GAN.
- The generator G of a styleGAN uses baseline proGAN architecture, and the size of generated images starts from 4*4 resolution to 1024*1024 resolution on the incremental addition of layers.
- Bi-linear up/down-sampling is used in both discriminator and generator.
- Introduction of mapping network, which is used to transform the input latent vector space into an intermediate vector space (w). This process is executed to disentangle the Style and Content features. The mapping function is implemented using 8-layers Multi-layer Perceptron (8 FC in Fig 2).
- The output of the mapping network is passed through a learned Affine transformation (A), that transforms intermediate latent space w to styles $y=(ys, yb)$ that controls the Adaptive Instance Normalization module of the synthesis network. This style vector gives control to the style of the generated image.

- The input to the AdaIN is $y = (y_s, y_b)$ which is generated by applying Affine transformation to the output of the mapping network. The AdaIN operation defined as the following equation:

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

Each feature map \mathbf{x}_i is normalized separately and scaled using the scalar y_s component and biased using the scalar y_b component of y . The synthesis network contains 18 convolutional layers 2 for each of the resolutions for $4 \times 4 - 1024 \times 1024$ resolution images. So, a total of 18 layers are present.

- The input is a constant matrix of $4 \times 4 \times 512$ dimension. Rather than taking a point from the latent space as input, there are two sources of randomness induced in generating the images. These results were extracted from the mapping network and the noise layers. The noise vector introduces stochastic variations in the generated image.

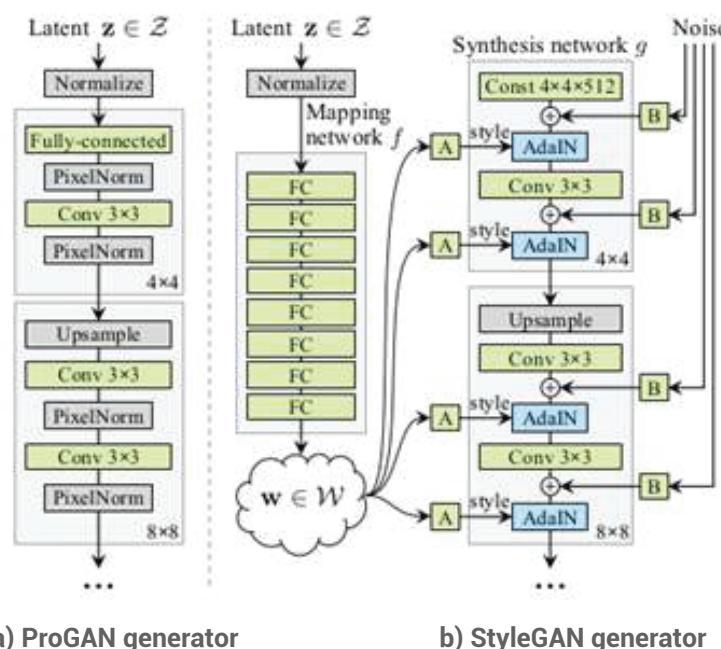


Fig 2. a-ProGAN generator, b-StyleGAN generator [Karras et al., 2019]

Following is the table depicting different configuration setups on which style GANs can be trained.

| Sl. No | Configuration |
|--------|--|
| 1 | 'config-a', # Baseline Progressive GANs |
| 2 | 'config-b', #+ Tuning (invl. bilinear up/down) |
| 3 | 'config-c', #+ Add mapping and styles |
| 4 | 'config-d', #+ Remove traditional input |
| 5 | 'config-e', #+ Add noise inputs |
| 6 | 'config-f', #+ Mixing regularization |

Table 1: Configuration for StyleGANs

Business objectives and possible solutions

| Objectives | Solutions |
|---|--|
| How many fake MK characters can be created in real-time? | Once the model is trained using the trained model, any number of fake images can be created. The training results using styleGANs on MK characters are shown in Fig 4, and fakes generated in real-time are shown in Fig5. |
| Can we control the style of images generated? (Instance: Can we generate only Male MK characters with beard and moustache?) | Yes, we can control the style of images generated by performing controlled synthesis of images using Conditional GANs, refer to Fig 4. |
| Can we mix the styles of 2 MK characters? | The latent vector representations of two test time images can be morphed together. The number of features from both images can be controlled by weight parameterization. The results of mixing two MK characters are shown in Fig 6. |
| Can we impose style features on general MK characters? (Instance: Can we change the hair/smile of a generated character?) | We can impose style features on generated MK characters by using trained vectors of hair, smile, gender, etc. Once the features are obtained, we can use style mixing to mix the test image and the feature. |

Experimental setup

Dataset

| Sl. No. | Dataset | Description | Images dimension used in model | Process for cleaning and aligning | Source |
|---------|--------------------|---|--------------------------------|--|--|
| 1 | Mortal Kompat (MK) | 20,000 high quality images retrieved from MK 11 game video of 3hours50minutes duration. 1028P HD movie.No. of the characters present are 25 | 256*256 | MTCNN-For face detection, cropping and alignment Perceptual Hashing from image De-duplicator | https://github.com/idealo/imagededup https://www.youtube.com/watch?v=CdEi4lr4IMQ&t=9878s |



Fig 3. Sample images from the training MK dataset

Experimental design

We have conducted a set of experiments to examine the performance of StyleGANs in terms of FID, quality of output produced, training time vs performance on FID. In addition, we also checked the results imposing pre-trained latent vectors on new faces of data and Mortal Kombat characters data. We have implemented GANs and performed its feasibility analysis to overcome the following issues:

- How effective StyleGANs are in producing MK characters with lesser data, low- resolution images, lighter architecture, and less training?
- How computationally extensive GANs are? Are GANs expensive to train? How to estimate the time and computational resources required to generate the desired output?
- How well the pre-trained vectors used for style-mixing to the original images?

Evaluation metrics

FID [Heusel et al. 2017]-Frechet Inception Distance score (FID) is a metric for image generation quality that calculates the distance between feature vectors calculated for real and generated images. FID is used to understand the quality of the image generated, the lower the FID score, higher the quality of image generated. The perfect FID score is zero. Experimental platform- All experiments are performed on the AWS platform, and the following is its configuration.

| Name | GPUs | vCPU | Memory (GiB) | GPU Memory (GiB) | Price/hr* (Linux) |
|------------|------|------|--------------|------------------|-------------------|
| g3.4xlarge | 1 | 16 | 122 | 8 | \$1.14 |

Experiments Results

- i. **Training** - The training images retrieved at a different number of kims are depicted in the following figures. All experiments were conducted at configuration "d". Refer to Table 1.



Fig 4. MK training results 1-Kims 2364, 2-Kims 5306 , 3-Kims 6126, 4-,Kims 8006 5-Kims 9766, 6- 10000 Kims

- ii. **Training time** - The following table depicts the time taken, and FID results for the model trained till 10000 kims. The best-trained models have an FID of 4.88 on the FFHQ dataset (Karras et al., 2019) on configuration d (refer table 1). We got the final FID score of 53.39 on the MK dataset.

| MK dataset | | | |
|------------|--------------------|------------------------------|--------|
| S.No. | Pickle file (Kims) | Time Taken (approximate) | FID |
| 1 | 140 | - | 396.47 |
| 2 | 2364 | 8 hours 15 minutes | 251.63 |
| 3 | 4085 | 1 day 4hours 18 minutes | 166.39 |
| 4 | 5306 | 1 day, 23 hours, 44 minutes | 57.97 |
| 5 | 6126 | 2 days, 13 hours, 15 minutes | 75.87 |
| 6 | 6926 | 2 days, 13 hours, 41 minutes | 60.30 |
| 7 | 7726 | 2 days, 14 hours, 4 minutes | 56.80 |
| 8 | 8526 | 2 days, 13 hours, 47 minutes | 55.02 |
| 9 | 9326 | 2 days, 13 hours, 56 minutes | 54.70 |
| 10 | 10000 | 2 days, 4 hours, 24 minutes | 53.39 |

Table 3. Time taken for training & FID score

- iii. **Real-time image generation** – Following are some of the fake images generated using different trained models using random seed values.



Fig 5. Real-time images generation (using seed value) @7726 kims pickle MK.

- iv. **Style Mixing** – Using the Style Mixing method to generate new variations in the character appearance where we could use two or more reference images to generate new results. In this context, we have used two characters to create a new character.



Fig 6. Style Mixing results on MK characters

- v. **Progressively growing GAN- results on MK** - When progressively growing GAN is applied to MK characters, with its nature of incrementally adding the layers to the network, the network learns to generate smaller and low-resolution images, followed by more complex images. Thus, stabilizing the training and overcoming mode collapse of the generators.

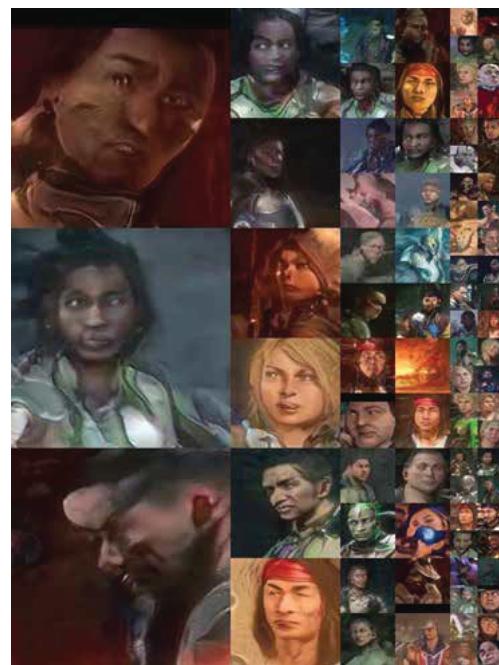


Fig 7. Progressive GANs output

Conclusion

In this whitepaper, we have discussed the methodology of applying the feasibility analysis using styleGANs for Mortal Kombat characters. We have provided a detailed report on the GANs types and evolution with respect to image generation approaches to solve different use cases using GANs. Which also includes style representation and conditional generation.

The latter part of our attempt provides the results of GANs training time, FID scores, real-time image generation output, style mixing, and ProGANs results.

After completing ~15 days of training (with GPU cost \$1.14 per hour), the FID score achieved is 53.39. The quality of images is also improved, which can be further enhanced by conducting flawless training sessions.

Recent advancements in GANs illustrate that a better GAN performance is achievable even with fewer data. Adaptive Discriminator Augmentation [Karras et al., 2020] and Differentiable Augmentation [Zhao et al., 2020] are a few of the recent approaches which have been proposed to train GANs effectively even with less amount of data, which is currently being researched in our CoE team.

References

- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).
- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Advances in neural information processing systems (pp. 6626-6637).
- Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196.
- Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4401-4410).
- Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., & Aila, T. (2020). Training generative adversarial networks with limited data. arXiv preprint arXiv:2006.06676.
- Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.
- Zhao, S., Liu, Z., Lin, J., Zhu, J. Y., & Han, S. (2020). Differentiable Augmentation for Data-Efficient GAN Training. arXiv preprint arXiv:2006.10738.



About Affine

Affine is a leading professional services & solutions firm, enabling global enterprises to affect their transformation & innovation, leveraging the unique Trifecta of AI, Data-Engineering & Cloud. With a globally distributed team of 500 + analytics professionals, Affine covers end-to-end capabilities spanning modern data engineering to core AI and scalable cloud deployment across North America, Europe, and Asia.

Affine combines the hyper-convergence of AI, Data-Engineering & Cloud, with its deep industry knowledge, particularly in Manufacturing, Gaming, CPG, and Technology segments. Affine demonstrates thought leadership in all relevant knowledge vectors by investing heavily in research through its highly acknowledged Centres-of-Excellence and strong academia relationships with reputed institutions like UC Berkeley and premier IITs in India.

Talk to our experts!

Discover how Affine can support your business transformation journey.

 www.affine.ai

 info@affine.ai