JOURNAL OF AWARENESS
HOLISTENCE publications

Journal of Awareness
Volume / Cilt: 8, Issue / Sayı: 1, 2023, pp.57-61
E-ISSN: 2149-6544
https://journals.gen.tr/joa
DOI: https://doi.org/10.26809/joa.1974

Received / Geliş: 20.12.2022
Accepted / Kabul: 31.01.2023

ARAŞTIRMA MAKALESİ/RESEARCH ARTICLE

Generating video game characters using StyleGAN2

İsmail Ergen

Asst.Prof. Faculty of Fine Arts, Design and Architecture, İstinye University, Türkiye, e-mail: ismailergen@gmail.comy

Abstract

GANs have been getting better and better each year. The state of the art GAN models for generating 2D images have become so good it is hard to differentiate generated images nowadays. In this paper we create 3 different sparse data sets from video game assets and train them with StyleGAN2 to generate new artwork based on the previously existing artworks of the video game in question

Keywords: Generative art, Videogame, StyleGAN

Citation/Atıf: ERGEN, İ. (2023). Generating video game characters using StyleGAN2. *Journal of Awareness*. 8(1): 57-61, DOI: 10.26809/joa.1974

Corresponding Author/ Sorumlu Yazar:
İsmail Ergen
E-mail: ismailergen@gmail.com

1. INTRODUCTION

In order to gain a place in the developing video game market, new video games are populated with assets that are eye-catching, iconic and unique to the game. These assets are created by artists, requiring lots of time and effort. There are all kinds of different types of artists created elements games use, but in this paper, we will focus on 2D art. For high number of video games, the work that goes into creating new 2D art is dependent on the existing art of the game since video games aim putting together a consistent virtual experience. Therefore, existing assets' style, form, color palette etc. have a strong impact on the assets created after. In this paper, we propose three new data sets, each of them consisting around 1K-2K images. And by training the state of the art GAN model StyleGAN2 [2] we produce new 2D assets from these sparse and highly limited data sets. We show that these newly produced images, might assist the creative thinking of artists or in some cases replace the artists altogether.

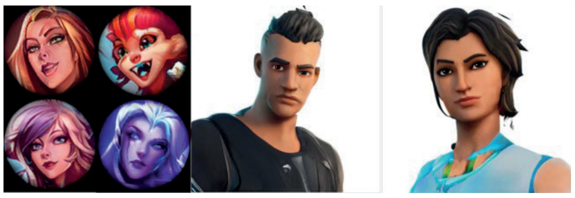


Figure 1. Example generated 2D assets, on left League of Legends character avatars and on right Fortnite character outfits.

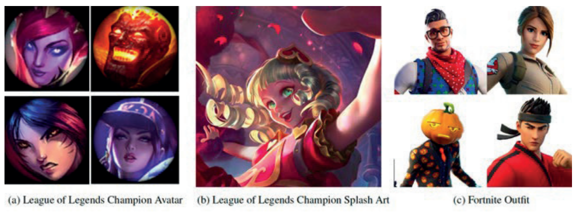
2. DATA COLLECTION AND PROCESSING

2.1. League of Legends Champion Avatar Icons

The first data set that was collected was League of Legends champion avatar icons. To obtain the data, 1289 avatars were downloaded by using the developer api https://developer.riotgames.com/ docs/lol. These avatars were made out of every champion skin in League of Legends. The League of Legends champions are made up from lots of different characters that can be described as humanoid, animal like, spiritual being etc. Each of these retrieved avatar images were of size 120x120. Since StyleGAN2 requires the input images to be of size 2 nx2 n and for the transfer learning the minimum FFHQ [3] trained model was on 256x256, these images were resized using the ImageMagick [5] utility into 256x256. How the dataset looked after these operations can be observed at Figure 2a.

2.2. League of Legends Champion Splash Art

The second data set that was collected was League of Legends champion splash arts. Similarly, to League of Legends champion avatar icons, these were obtained at https://developer.riotgames.com/docs/ lol using Leauge of Legends developer API. There was 1289 of them too just like avatars.


(a) League of Legends Champion Avatar
(b) League of Legends Champion Splash Art
(c) Fortnite Outfit
Figure 2. Retrieved and processed data set samples.

Splash art images were of size 1215x717. Each of these images consists a big background that is specific to the character. Since our focus is on the character here, it was needed to cut these images around the characters. To achieve this cut around the character, it was decided that a window size of 512x512 was going to be used. And to center this window around the character two approaches were used.

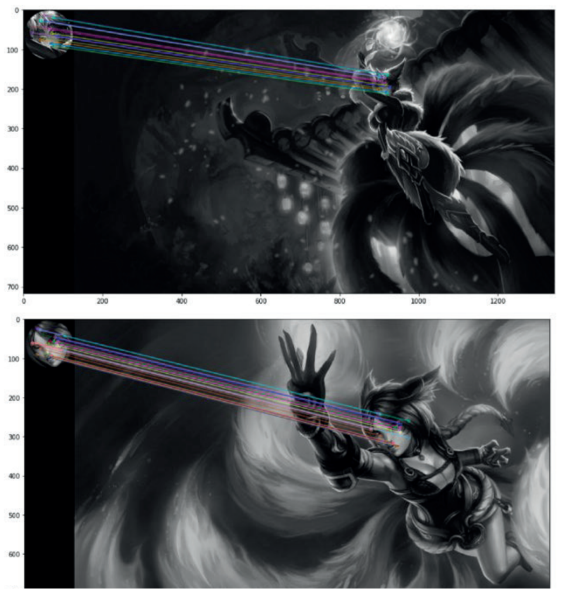


Figure 3. Avatars getting feature matched to a corresponding splash image

First approach was downloading another set of champion images called loading images. These images are basically cut-outs that focus around the whole body of the character. An example of loading image can be seen in Figure 4. After acquiring these loading images, OpenCV [1] library was used to template match all of these loading images to their corresponding splash images to find a character body center point in splash images. And using the center point values, the splash image was cut in the defined window size around the center point. Doing a relatively similarly centered dataset, but for some of the images template matching failed finding the center point. These images were removed from the data set and although the remaining images looked well centered their faces didn't align with the other images in the dataset most of the time.

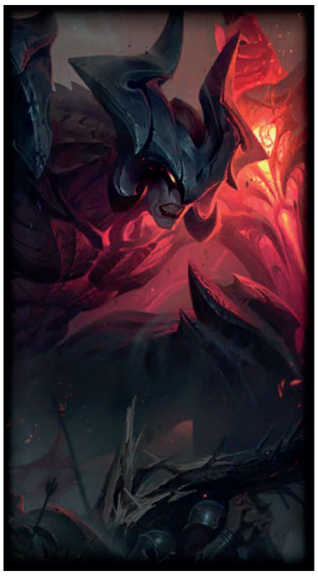


Figure 4. League of Legends Champion Loading Image.

Since the template matching with loading images, failed centering the faces. A second approach was tried this time finding the center point by using champion avatars. Since avatars had a circular cutout around them, and there were some effects applied to the avatar borders. Scaling them to appropriate sizes and trying to apply template matching against splash images immediately failed. So instead of template matching a more sophisticated method called feature matching was used and it succeed. Examples of an avatar getting feature matched to the splash image can be seen on Figure 3.

Using the center from the second approach images were cut to the size of 512x512. Producing a mostly face centered, character image dataset that can be seen at Figure 2b.

2.3. Fortnite Outfit Icons

The last data set that was prepared was made up from Fortnite outfit icons. A total of 1267 images were fetched from the website https://skindb.co/fortnite. Each image that was fetched was already of size 512x512 so

scaling wasn't necessary. But color space of these images had an alpha layer so that was removed and anywhere that was transparent were filled with white for the whole dataset. Samples from the resulting dataset can be seen at Figure 2c.

3. TRAINING

For the training of the model Google Colab environment was picked. And the state of the art GAN model Style - GAN2 [2] with ADA(Adaptive Discriminator Augmentation) [4] was used. For all of the trained model instances transfer learning was applied on top of the models pretrained with FFHQ [3].

While deciding on the training parameters only gamma value (Gamma) and augs (Augmentations Specifier) were changed between trainings. Also, since the training of the model with the GPU resources from Google Colab already took quite a bit as is, instead of using a quantitative quality metric like fid50k full (which almost tripled training times, adding 45 minutes between each snapshot) the quality of the model's situation was done qualitatively. Specifically,

Table 1. Average timings for Tesla P100 GPU on Google Colab.

Dataset Image Resolution	sec/tick	sec/kimg
128x128	144	38
256x256	267	67
512x512	1083	270

by observing set of outputs with constant seeds between snapshots and traversing incrementally on the feature planes to check for signs of over fitting.

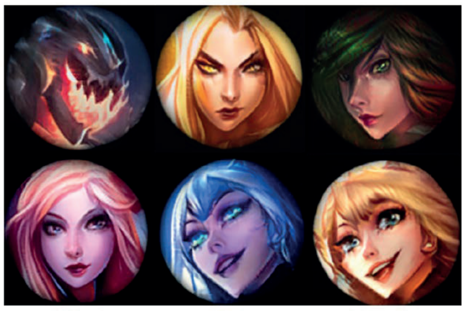
For each of the datasets mirroring on x axis was applied and the best performing augmentation setting was found to be as ’bg’, meaning pixel blitting, geometric transforms only. According to Karras et al. [4] for a dataset of 2k images only using pixel blitting, geometric and color transforms yielded best results. Which is a quite similar consensus to ours. And for the gamma, it was settled upon the value of 50.

The training went at different speeds for different image resolutions which can be observed on Table 1. Be aware that here speeds are averaged for Tesla P100 GPUs running on Google Colab.

Respectively we were able to train data sets League of Legends champion avatars for around 2000 kimg, League of Legends champion splash arts for 1600 kimg and Fortnite outfit icons for around 1000 kimg.

4. RESULTS

Looking at the outputs of each trained model both handpicked at Figure 5 and random at Figure 6, it is obvious that the worst performing one is the League of Legends champion splash arts based one. And it seems like the Fortnite outfit icons is the best performing one even though it didn’t have the time to train as much as others. It looks like the reason behind Fortnite one being the best is the data set being way cleaner, all the faces are almost aligned at the same place and the background is plain white.


(a) League of Legends champion avatars

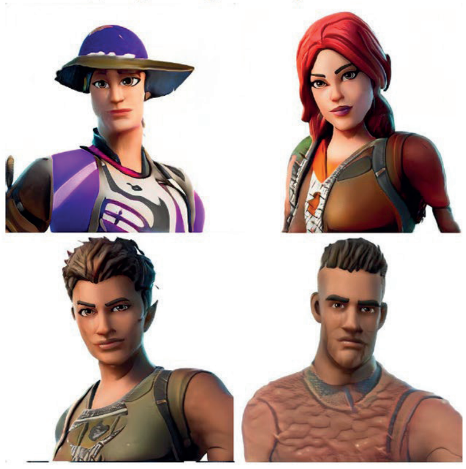
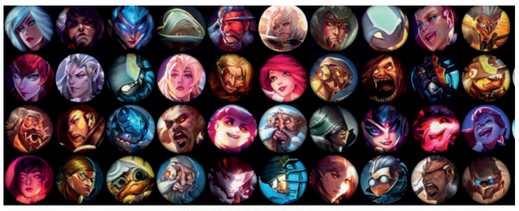
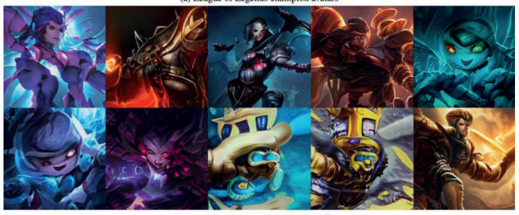

(b) Fortnite outfit icons

Figure 5. Handpicked examples for trained models.


(a) League of Legends champion avatars


(b) League of Legends champion splash arts

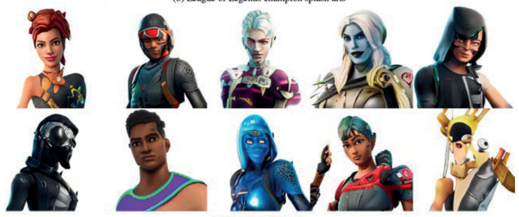

(c) Fortnite outfit icons

Figure 6. Random examples for each trained model.

5. CONCLUSION

Looking at the results it is clear that, even if the datasets that are being used are not large, and very much sparse, it is possible to train a model that can have some convincing outputs. Even if the outputs are not ready to publish in a video game, they are certainly close. Also it is pretty possible for an artist to use these outputs when in need of inspiration. Looking at the amount of progress GANs had in recent years, it reasonable to guess that in near future artists will incorporate GANs in their work even more so than now.

REFERENCES

BRADISKI, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.

Karras et al. Analyzing and improving the image quality of StyleGAN.

Karras et al. A style-based generator architecture for generative adversarial networks.

Karras et al. Training generative adversarial networks with limited data.

The ImageMagick Development Team. Imagemagick.