Using GANs for Sprite Generation

Gavin Thomas

Denison University

thomas_g8@denison.edu

This paper proposes the use of a Generative Adversarial Network (GAN) to generate novel images of video game sprite characters. The generation of unique characters in video games could make the gaming experience unique to each individual playing the game which could decrease potential boredom due to repetition. GANs are composed of two adversarial models, the generator and the discriminator, who compete against each other in a zero sum game until Nash's equilibrium is reached or training is terminated. The GAN in this paper has a standard architecture. In this paper, the discriminator is a convolutional neural network and the generator is a transposed convolutional network. This model was trained on 4,272 character images collected from Open Game Art using web scraping. In total, this dataset contains 16,627 CC0 licensed images of sprites from 284 collections. These images were manually sorted into 5 categories: Characters, Items, Animated, Vehicles, and Tokens/Icons.

Because the data was scraped from 284 collections, there were two major data impurities: similar images and different image sizes. As the sprites are meant for video games, many artists provided their character sprites doing various actions, e.g. swinging a sword. To not overfit to these character sprites, similar images were removed from the training set using a structural similarity index calculation. Images in each collection were converted to a network, with connections being the similarity index between two images. Then images were iteratively removed with similarity connections greater than 0.9 out of 1.0. To fit to the 32x32x3 size requirement for the GAN, many images required major resizing. To avoid loss of detail due to downsizing, the images were first cropped using the Canny edge detection algorithm. Then the images were downsized using the inter area interpolation method.

These images were passed through a conventional generator and discriminator architecture. The objective creativity of the generated sprites was quantitatively analyzed with a survey of 40 participants. The results showed that most participants were not able to accurately distinguish between human and computer generated sprites. In addition, it shows that the participants were inherently distrustful of all images, classifying significantly more images as computer generated, as opposed to human generated.

I. Introduction

Generative Adversarial Networks (GANs) were first proposed in 2014 [8]. GANs can mimic any distribution of data, allowing for the creation of novel images, music, text, and other related data. GANs are built on two adversarial models: the generator G and discriminator D, which compete against each other in a zero-sum game [8]. The generator can be likened to an art forger trying to trick an art expert, or the discriminator, into thinking the work it created is real. Recently, the quality of images produced by GANs has been rapidly improving, although the inner processes are still hidden by the infamous black box of machine learning [3, 12].

In the genre of art and image generation, recent work has focused on generating two dimensional images, from paintings to realistic photographs [7, 11]. In 2018, the first computer generated artwork, created by a GAN, was even sold at Christie's auction for $432,500 [4]. However, there has been little work generating video game items and characters; moreover, the work that has been done in this domain uses limited datasets or suffers from other limitations [10, 25]. If done successfully, artificially generated characters would enhance user's gaming experience by reducing the repetitive nature of enemies in games. These generated character's could be different for each player, making their experience with the game truly unique.

This paper will address this gap and generate video game character sprites, using a novel dataset collected from Open Game Art and a conventional GAN architecture[1] [1]. This architecture does not modify the generator, discriminator, or loss function and follows recent work in optimizing GAN performance [3, 15, 17, 20]. Although it can be argued that no GAN is truly creative, my goal is to simulate creativity and make my output indistinguishable from human created sprites. This creativity will be measured in a survey where participants are asked to distinguish human vs computer generated art.

This paper also uses a new data cleaning method. A subset of the images are taken by eliminating images that are too structurally similar to other images to avoid overfitting.

[1] Architecture refers to the arrangement of layers and the connection between layers in a neural network.

In addition, due to the large difference in image shape, the images are cropped, using edge detection, prior to the resizing process.

In addition, I present a new dataset of 23,672 video game sprite images (opengameart-2Dsprites) that provides a diverse set of public domain images. While the images are not new, they have never been collected into a single dataset. This dataset is publicly available and included with the source code and includes attribution to the collections from which the images were taken.

II. Domain Review

The capability of GANs for image generation has led to some fascinating research. GANs have been used to generate realistic looking eyes, faces, and even Anime characters as shown in Figure 1 [11, 12, 21].

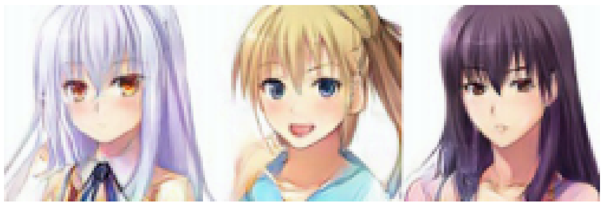


Figure 1: Generated Anime Characters using DRAGAN + attribute tags [11].

One of the biggest challenges of implementing Generative Adversarial Networks is stabilizing the training process. To address this issue, the DCGAN, which adds additional architectural constraints, was developed [19]. These constraints are: replace pooling layers with strided convolutions, eliminate fully connected layers on top of convolutional layers, use batch normalization in both models. Batch normalization is a technique where the inputs of each layer are normalized so the standard deviation is one and the mean output activation is zero [19]. This process prevents the generator from collapsing all samples to a single point and helps gradient flow, especially in particularly sophisticated models.

The authors used their model to generate bedrooms with five epochs of training and provided samples of their generated rooms. For empirical results, the authors used the DCGAN as a feature extractor on CIFAR-10 (but trained on Imagenet-1K) and reported an accuracy of 73.8 percent. They did not provide other evaluation numbers, like the inception score or FID, because these methods had not been produced yet [19]. This paper highlights the importance of conducting experiments after training the model to understand the latent space and peer into the black box of convolutional neural networks.

Building off the previously proposed DCGAN, Lewis Horsley and Diego Perez-Liebana created an automatic sprite generator [10]. In their work, they focused on creating sprites from human-like characters, creatures, and faces datasets. These datasets had 1210, 36, and 517 images for training, respectively. They state in their conclusion that the tiny and diverse dataset severely limited their analysis [10]. Figure 2 shows the generated creatures after 15000 epochs.

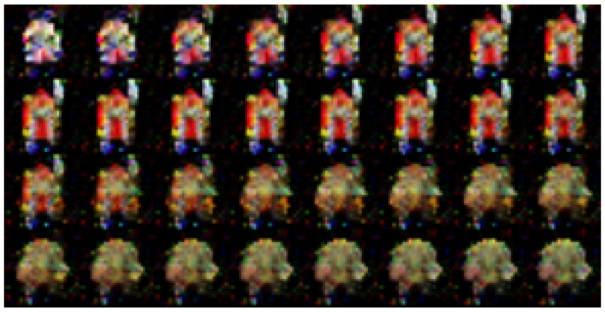


Figure 2: Generated DCGAN Characters [10]

Another critique of GANs is that they are emulative and not creative. To address this gap, the Creative Adversarial Network was proposed in 2017 [7]. They argued that standard GANs have a limited ability to produce creative art, but by maximizing deviation from established styles and minimizing deviation from the art distribution, the Creative Adversarial Network can create images indistinguishable from human generated art to observers. This Creative Adversarial Network was trained on data gathered from WikiArt, consisting of 81,449 publicly available paintings with 25 different styles.

To evaluate the created work, the authors compared their results to variants of the DCGAN and showed that the CAN model better emulate the art distribution [7]. Next, they evaluated the creativity of their model through a series of survey experiments and quantitatively showed their images were indistinguishable from human-generated work; however, this could be due to the ambiguity of what is considered art in the modern era.

From art, another creative domain that GANs have been implemented is the generation of new clothes on an individual using the DeepFashion dataset [26]. In this study, the authors included a sentence describing an outfit as an additional input to their model with an image of an individual. Then, after generating a segmentation map (contains pixel-wise labels of features e.g. hair) and their newly proposed compositional mapping layer, the article of clothing is rendered on the individual in the original image. As clothes are subjective to the individual, similarly to the aforementioned Creative Adversarial Network, the authors quantitatively supported their results by conducting a survey of 50 individuals to rank the quality of the generated images.

How to quantitatively evaluate the results of generative models is still up for debate. The lack of accepted performance metrics for unsupervised learning, and specifically GANs, poses a challenge for comparing different models. In 2016, utilization of an inception score was proposed [20]. They argue that the inception score is strongly correlated with human judgement, avoiding potential biases that arise with surveying but requiring a sufficient number of samples [20]. The inception score is based off the Inception model which is a widely used recognition model Inception-v3 pre-trained on ImageNet [24].

More recently, the Fréchet Inception Distance (FID) has been introduced to improve upon the inception score by comparing the statistics of generated samples to those of real samples [9]. Other authors use a simple survey of human participants to analyze the performance of their generated images; however, this can be subject to various biases if not conducted properly.

III. Methods

A. Generative Adversarial Networks

A GAN is composed of two models, the Generator G and the Discriminator D, who face off in a zero-sum game with the generator trying to fool the discriminator [8]. Figure 3 shows the architecture of a basic GAN. As depicted, the generator is directly linked to the discriminator, as the output of the generator becomes the input for the discriminator; furthermore, the discriminator's classification is used by the generator to update its weights through backpropagation. Generally, discriminative models probabilistically classify input data, mapping features to labels. Conversely, generative models determine the probability of the feature given a label.

In the traditional GAN architecture, the generative model creates data to be inputted into the discriminative model, which determines if the generated data is from the true distribution. Both models get better and better at their respective roles over training until the Nash equilibruim[2] is reached or training is terminated [20]. This type of training is defined as adversarial training.

The following equation shows the value function $V(G, D)$ of the minimax game that is played by the discriminator and the generator during the training process:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim\ p_{\text{data}}}[\log D(x)] \\ +\mathbb{E}_{z\sim\ p_{\text{z}}}[\log(1 - D((G(z)))]$$

where z is the random noise vector that is sampled from the p_z distribution and x is an image from the p_{data} distribution [8].

Breaking this down, the generator $G(z, \theta)$ aims to generate new data similar to the real

[2] The Nash equilibrium is a stable state where no participant can gain by change of strategies.

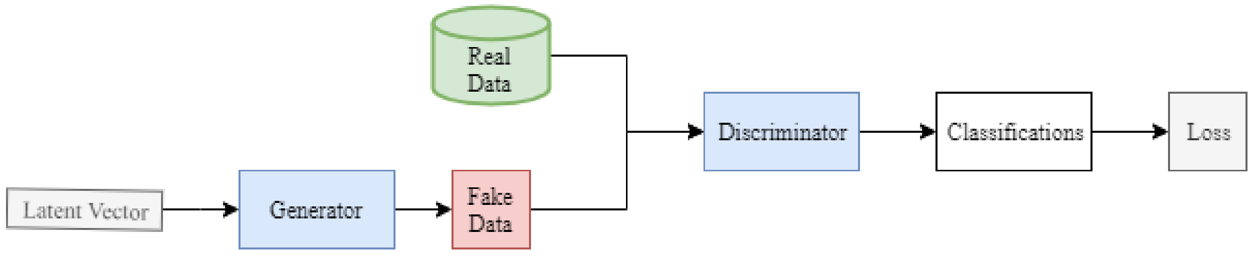


Figure 3: GAN Architecture

sample. The generator maps input noise z to the desired $_{\text{data}}$ distribution, which, in the case of this paper, is the sprite image data distribution. In this paper, where the generator is a transposed convolutional network, θ represents the weights in the network.

Next, the discriminator $D(x, \theta)$ aims to classify the input data (consisting of images generated by the discriminator and the real sample) as real or generated. As stated above, x is an image from the real image dataset while θ represents the weights in the network. More technically, the goal of the discriminator is to minimize the probability of classifying a generated image as real.

For the purposes of this paper, the generator and discriminator are optimized alternatively for every batch. This alternating training process allows for the generator to create images that better emulate the training distribution.

Thus, to effectively train a GAN, one must:

1. Sample the noise and real datasets, each with size n
2. Use this data to train the Discriminator
3. Sample another noise subset with the same size n as in step 1
4. Use this data to train the Generator
5. Repeat until the the Nash equilibruim is reached or training is terminated

B. Background

In the context of this paper, the discriminator is a Convolutional Neural Network (CNN), while the generator is a Transposed Convolutional Neural Network. CNNs, similarly to deep neural networks, are comprised of neurons with weights and bases; however, if one was to use a neural network on a simple image, the number of neurons would quickly rise to the millions and computation would become extremely arduous [14]. To be more computationally efficient, CNNs are restricted by their local spatiality to reduce the number of parameters in the network decomposed into layers; each layer has a width

W, height H, and depth (color channel) D.

In this review, I will briefly elucidate the main layers and elements of the GAN architecture that I will use in my analysis. These are convolutional layers, transposed convolutional layers, fully connected layers, leaky rectified linear units, and the Adam optimization algorithm.

1. Convolutional Layer

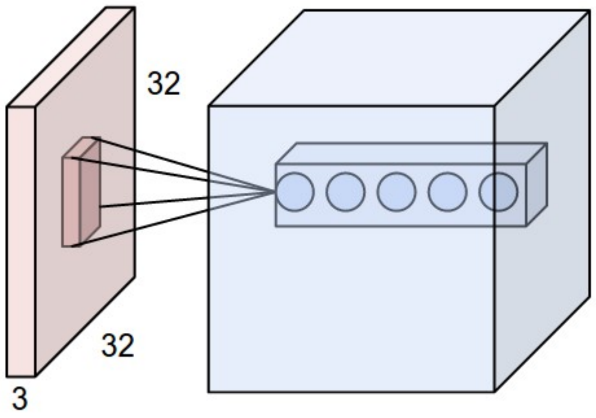


Figure 4: Example First Convolutional Layer

Convolutional layers are traditionally the first layer in a CNN. A convolutional layer tries to learn the features of the image using small squares of input data using the RELU activation function. It takes an input of $W1 * H1 * D1$ and contains the hyperparameters K, F, S, and P, which stand for number of filters, spatial extent, stride, and amount of zero padding, respectively. There are $F * F * D1$ weights per filter. The output of the convolutional layer is $W2 * H2 * D2$, where the size of these layers is determined by the following [14]:

$$W2 = (W1 - F + 2P)/(S + 1)$$

$$H2 = (H1 - F + 2P)/(S + 1)$$

$$D2 = K$$

S, or stride, determines how many pixels the filters move at a time. The larger the stride, the smaller the output will be, as demonstrated by the equations above. Figure 4 depicts a hypothetical first convolutional layer in a model [14]. Recently, in practice, larger strides are used to act as pooling layers, which reduces the spatial size and number of parameters, thus controlling overfitting without decreasing accuracy [22].

2. Transposed Convolutional Layer

A Transposed Convolutional Layer, or Deconvolutional Layer, acts as the opposite of a convolutional layer. Simply put, a transposed convolutional layer performs an upsampling operation and interprets the data to fill in detail, causing the output to grow rather than shrink (contingent on parameters used) [6, 18]. These layers are used to transform data in the opposite direction of a normal convolutional layer in the construction of images. These layers are most commonly found in the generative model in a traditional GAN architecure.

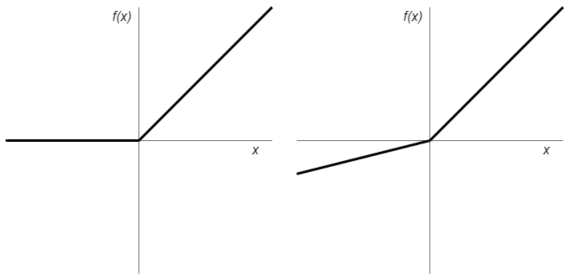
3. Fully Connected Layer

The fully connected layer flattens the multi-dimensional image matrix into a vector and feeds it into a layer similar to a traditional neural network in the sense that it is comprised of nodes and activations that are totally connected to the previous layer. This translates the features learned by the convolutional layer into classifiable items [14]. Traditionally, one or more fully connected layer will be the final layers in the model.

Between these layers (and others) one can specify the dropout rate to help avoid overfitting [23]. This 'dropout' refers to ignoring certain neurons during the training process for both hidden and visible units. Probabilistically, with each training step, an individual node has a probability of $1 - p$ of being kept. It has been shown that dropout can also be applied to convolutional layers, but should be at much lower levels [18].

4. Leaky Rectified Linear Units

The Rectified Linear Units (ReLU) is a commonly used activation function used in a variety of deep learning models [2]. Formally, this is written as $f(x) = \max(0, x)$. Translated, the ReLU activation function returns 0 if x is negative or x if x is positive. A variation on this, called Leaky Rectified Linear Units (Leaky ReLU), allow for a small, non-zero gradient when the node is not active to avoid the "dead ReLU" problem which occurs with constant 0 gradients [16]. Formally, this is written as $f(x) = \max(\alpha * x, x)$. The difference between ReLU and Leaky ReLU can be easily discerned graphically in Figure 5. I use Leaky ReLU in the discriminator and the generator.



(a) ReLU (b) Leaky ReLU

Figure 5: Activation Functions

5. Adam Optimization Algorithm

The Adam Optimization Algorithm is a method for stochastic first-order gradient-based optimization [13]. The name Adam comes from its usage of adaptive moment estimation as the algorithm computes the adaptive learning rates for different parameters [13]. Unlike stochastic gradient descent, Adam maintains and changes a learning rate (α) for each parameter in the network. Adam has been shown to be computationally efficient and handles noisy or sparse gradients, making it extremely popular in recent studies [13].

C. Data

The data for this project was collected from Open Game Art (opengameart.org) [1]. Open Game Art is a repository providing freely licensed art from and for game developers. Using Python, I iteratively gathered collections from 284 different pages (collections) which are made by different artists who classify their work under the creative commons (CC0) license. The CC0 licence allows for individuals to use the images for any projects, including academic and commercial. In addition, because I web scraped this data, I ensured that the usage of a robot was allowed on the nodes that I accessed.

For each collection, I gathered every sprite image and standardized the names for each by using an index. This was required because many of the images had identical names. Overall, the collected data includes 23,672 images. I manually sorted these images into 5 categories as shown in Table I. Each sprite was assigned to an individual classification, while each collection could contain images of multiple classifications. 16,627 of the images fit neatly into these classifications, but 7,045 images were omitted. These omitted images were often background tiles, buildings, or trees.

Classification	Images	Collections
Characters	5962	105
Items	4797	97
Animated	990	79
Vehicles	1556	33
Tokens/Icon	3322	36

TABLE I: Categories

In this analysis, I will be focusing on the Characters class. The Characters dataset is incredibly diverse in color and shape of the characters. The images include human-like characters as well as more abstract monsters from multiple angles, which may blur the output. Figure 6 displays 20 sample images of the Character class that I will use in my analysis.

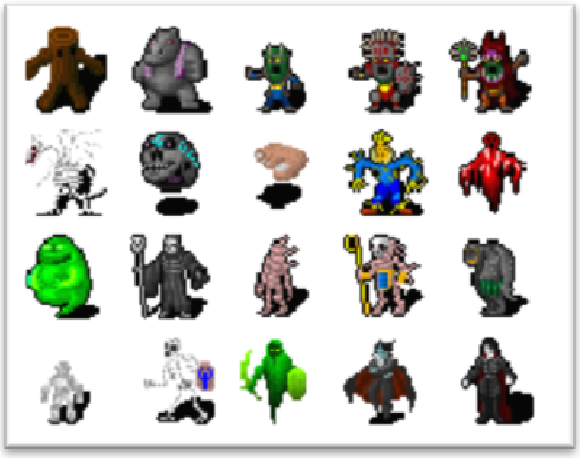
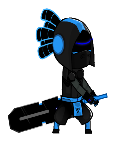
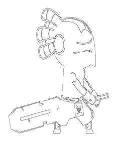
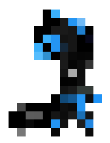


Figure 6: Example Sprites

Because there was a vast range in dimensions of the images, it was necessary that


(a) Original (1400x1099)


(b) Edge Detection (1400x1099)


(c) Vanilla Resize (32x32)

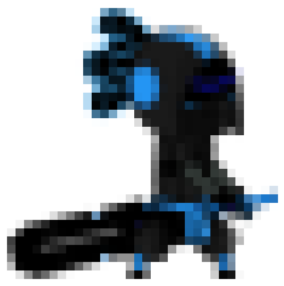

(d) Cropped Resize (32x32)

Figure 7: Resizing Process

I standardize the sizes of the images to be inputted into the GAN. I decided to resize each image to 32x32x3 to reduce computation time and still capture the colors of the sprites. The resizing process, however, initially caused major distortion in images with large backgrounds and/or non-centered sprites. This distortion can be seen when comparing the sprite in Figure 7a to Figure 7c. Figure 7c loses almost all the complexity and clarity of the original sprite.

To reduce this distortion, I hypothesized that initially cropping these images before size reduction would yield clearer final images and better standardize my data. To do this, I used the Canny edge detection algorithm, with threshold values of 1 plus or minus 0.33 times the median pixel value, to calculate the outer edges of the image. An example of the Canny edge detection algorithm on Figure 7a is shown in Figure 7b. The outer edges of the images were then used to crop the excess background from the image and center the sprite. Finally, I resized the images using the inter area interpolation method which uses pixel area relation. This process resulted in images that more accurately represented the original images, as shown in Figure 7d.

Another limitation of the data, beside resizing the images, was that there were large samples of very similar images. This is a product of the fact that many images showing individual sprite's movement animation, which show very small sprite movements such as swinging a sword, were included as individual images. To help avoid overfitting to these images, I calculated the mean structural similarity index, which calculates similarity using the images luminance, contrast, and structure, for the nearest 200 images in each collection. 200 was selected due to computational limits; the similarity calculation was limited to each collection per the hypothesis that no two artists would make highly similar images.

Then, for each collection, I created a network of images with similarity scores higher than 0.9 out of 1.0 and iteratively removed images that had the most connections until no similarity scores above 0.9 remained. This process removed 1690 images from the Characters class, resulting in 4,272 images for the analysis.

With my images cleaned, I created a simple Convolutional Neural Network (CNN) to classify the images into the 5 groups. The CNN consisted of 3 convolutional layers, followed by a flattening and then two fully connected layers. This model was able to predict 41 percent of the images correctly. This is potentially worrisome, as the images were manually sorted and could have overlaps between categories; however, not much weight was put into the output of this network.

IV. Results

A. Model Composition

1. Discriminator

The discriminator is comprised of 3 convolutional layers. Each of these layers has a kernel size of 5 and strides of 2. In addition, each of these layers used the Leaky ReLU activation function and a dropout rate of 0.4. The three convolutional layers use 64, 128, and 256 filters, respectively. These layers are followed by a flattening layer and fully connected layer.

This model is compiled using binary cross entropy loss and the Adam optimizer with a learning rate of 0.002 and a beta 1 value of 0.5. The discriminator architecture is visually shown in Figure 8.

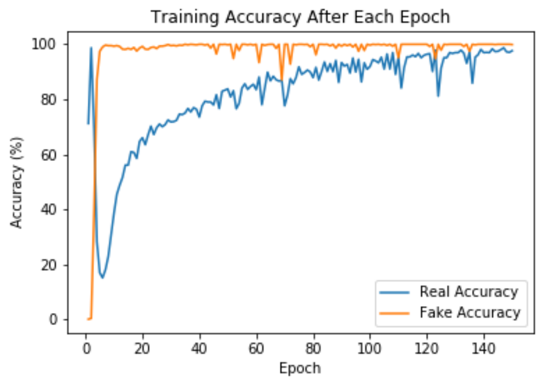


Figure 9: Discriminator Initial Training

To test the performance of my discriminator before running the entire GAN, I used samples of 1024 images over 150 epochs and achieved 99 percent accurate predictions for both real and fake images, as shown in Figure 9. This is indicative that the model trained successfully and is on par with the initial discriminator accuracy for comparable models.

2. Generator

The first layer in the generator is a fully connected layer. It takes in 4096 nodes, a latent space input dimension of 100, and uses

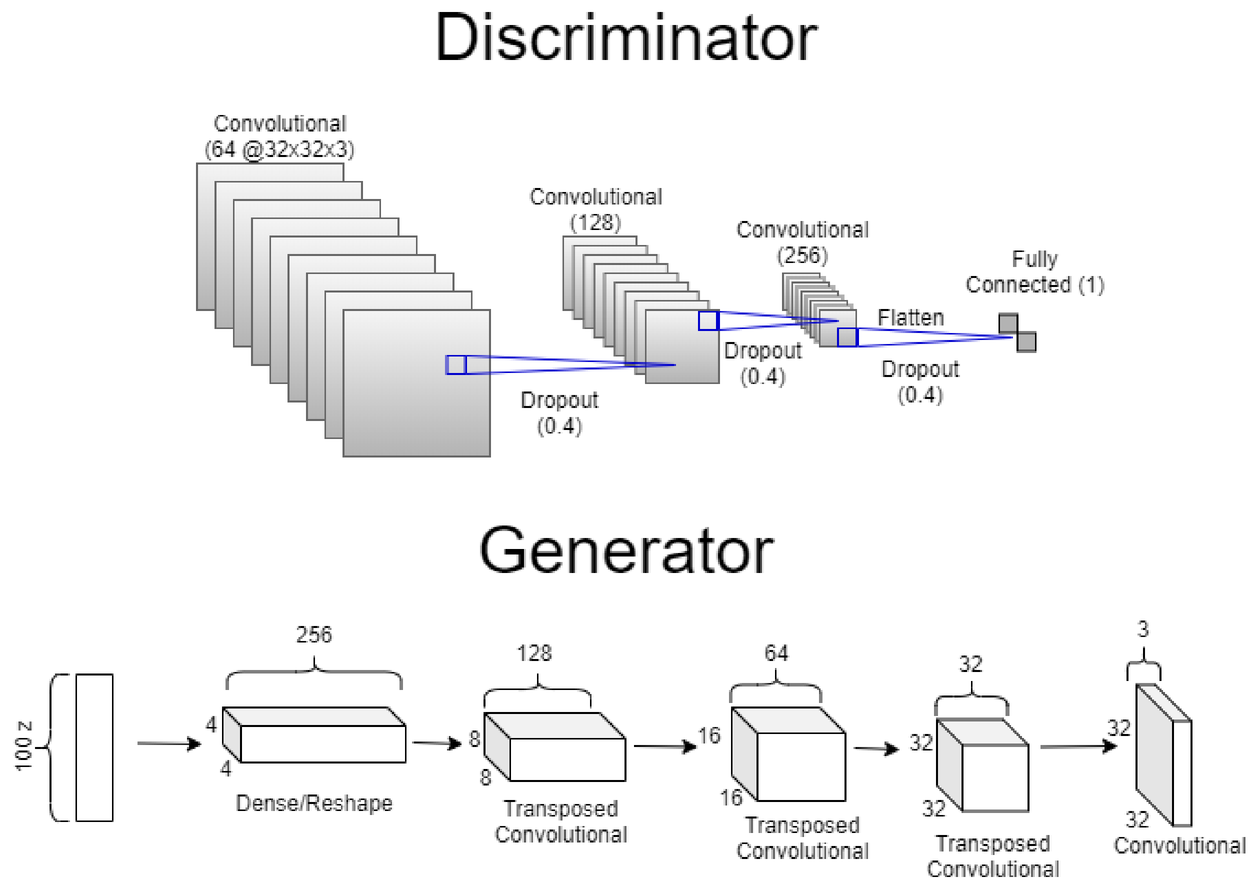


Figure 8: GAN Architecture

the Leaky ReLU activation function. This layer is then reshaped to 4x4x256. Next, there are three subsequent transpositional convolutional layers. These layers have 128 filters, kernel sizes of 4, strides of 2, and use the Leaky ReLU activation function. These layers reshape the data to images of 8x8, 16x16, and 32x32 pixels, respectively. Finally, the generator has a convolutional layer with the hyperbolic tangent activation function. The discriminator architecture is visually shown in Figure 8.

The first iteration of the generator (used to train the initial discriminator), randomly assigns a value to each pixel. Figure 10 shows the initial generator's output.

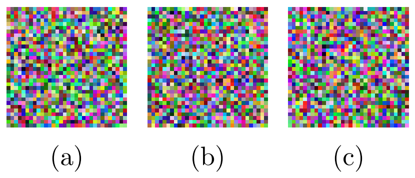


Figure 10: Initial Generator

B. Qualitative Evaluation

Figure 11 shows the improvement of the quality of generated sprite over time, with intervals of 100 epochs between each row.

Row 1, 2, 3, 4, 5, and 6 correspond to epoch 0, 100, 200, 300, 400, and 500, respectively. The 500 epochs took 34 hours to train. Figure 12 shows discriminator and generator loss for each epoch.

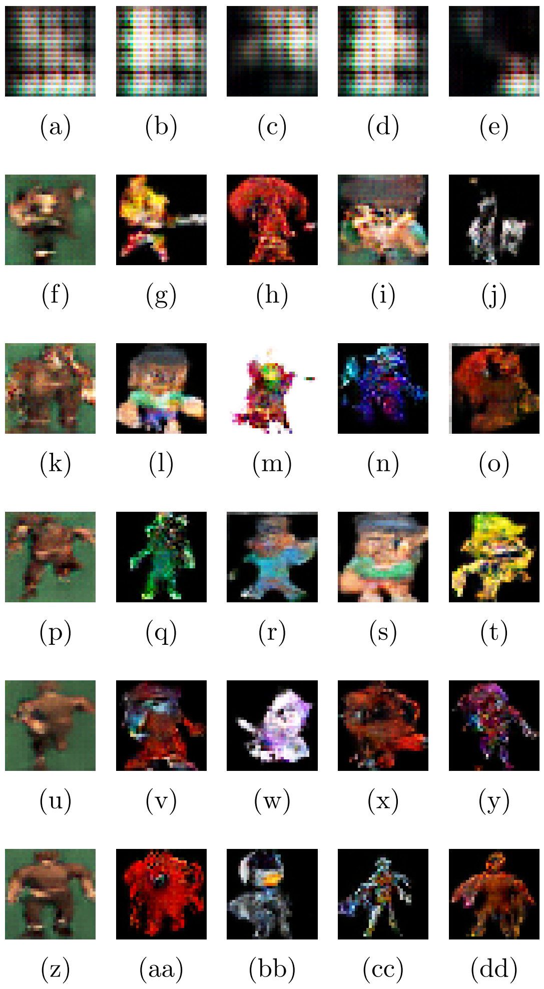


Figure 11: Generated Images by Epoch (Resized from 32x32x3)

The largest increase in quality of image can be seen between epoch 0 and 100. Clearly the generator beings to distinguish that each image needs a central body. The improvements in the subsequent epochs are much smaller. Slowly, limbs and eventually heads begin to appear. In the final model, some generated images (a, aa, bb) look highly similar to images in the dataset, but others (cc, dd) are novel images.

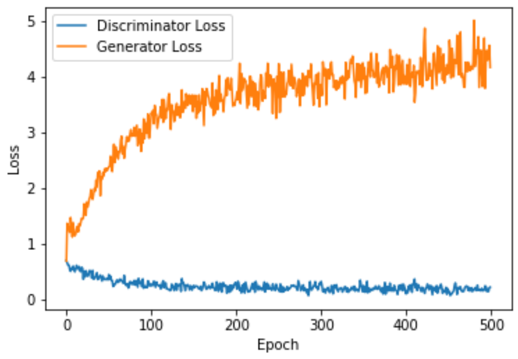


Figure 12: Discriminator and Generator Loss

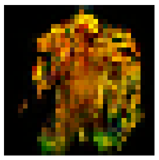


Figure 13: Poor Output

Not every image was clearly a character sprite however. Some images, like that shown in Figure 13, are highly pixelated, without the distinct features shown in Figure 11. Many of these poor sprites can be explained due to the high variability of poses and creatures in the training set. In addition, there seems to have been overfitting to some of the more common images in the dataset (see Figure 11z). This is likely due to the high variability in the images within the dataset; although, a blurry body can still be seen in the image.

C. Quantitative Evaluation

1. Survey Design

To quantifiably measure the quality of my output images, I used an online Qualtrics survey with 40 human participants[3]. The survey was shared via a survey link, given to the students and professors at Denison University. Although this sample could be classified as a convenience sample, I believe that is is representative of the population at large. This is because very few individuals can knowledgeably identity human vs computer generated images.

Confidentiality and anonymity were ensured because no personal information was collected from the participants. In Qualtrics, not even the IP addresses of the participants were collected. The results from each participant will be documented as a series of strings and stored for reproducibility.

Before beginning the survey, the participant was told that some of the images were human generated and others were computer generated. It is worth noting that the participants were not told how many images were included in each class. This could have caused an unintentional bias, as participants could inherently prefer to choose computer or human generated. This potential bias is unlikely to significantly invalidate the results and could actually provide meaningful insight.

The first page of the survey included the informed consent, which stated that by completing the survey, the participant agreed to participate in the survey. No deception will be involved as it will be clearly stated that the images will be generated by either a human or a computer. This survey was approved by the Denison University IRB board and classified as exempt; furthermore, upon the creation the survey, it was approved by the Denison University IRB chair.

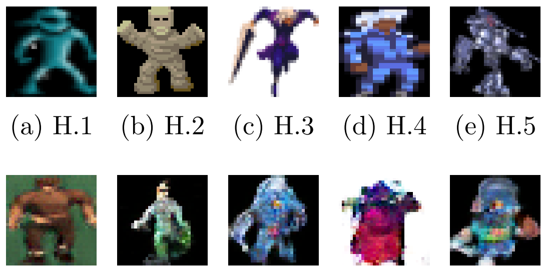


Figure 14: Survey Images where H=Human and C=Computer (Resized from 32x32x3)

Each participant was shown the ten images in 14 one at a time, some generated and others made by artists, and classified them as computer or human generated. For

[3] The data for this paper was generated using Qualtrics software, Version qualtrics XM of Qualtrics. Copyright © 2019 Qualtrics. Qualtrics and all other Qualtrics product or service names are registered trademarks or trademarks of Qualtrics, Provo, UT, USA. https://www.qualtrics.com

each image, the participant was asked, "Do you think the image shown was created by a human or a computer?" The participants were provided with a multiple choice answer of "Computer" or "Human". No responses were left blank.

2. Survey Results

As expected in the survey design, there was a bias towards classifying the images as computer generated. In the survey, the participants selected Computer 59 percent of the time (236/400). This could show that when told that some of the images were computer generated, the participants were inherently distrustful of all the images, and more likely to select computer generated. For the human generated images, located in the top row of Figure 14, participants correctly selected Human 46.5 percent of the time (93/200). Conversely, for the computer generated images, located in the bottom row of Figure 14, participants correctly selected Computer 64.5 percent of the time (129/200).

Within the computer generated images, the difference between the count of computer and human responses was not significant at the 5 percent level (p=0.396). Conversely, for the human generated images, the difference between count of computer and human responses was significant at the 5 percent level (p=0.0004). Upon running a Chi-Square test, I can conclude that there is a relationship between the true values and the survey responses at the 5 percent level (p=0.025). Finally, I reject the null that there is no trend at the 5 percent level per my Cochran-Armitage test for trend (p=0.020). This leads me believe that participants were able to select the computer generated images as computer generated with more confidence than selecting human generated as human generated.

Image	Computer	Human
Human 1	30	10
Human 2	13	27
Human 3	17	23
Human 4	24	16
Human 5	23	17
Human Total	106	94

Image	Computer	Human
Computer 1	23	17
Computer 2	25	15
Computer 3	28	12
Computer 4	26	14
Computer 5	26	14
Computer Total	129	71

TABLE II: Survey Responses

The mean score, as defined as number correct out of 10, in the survey was 5.525 and had a standard deviation of 1.87. As shown on Figure 15, only one out of the 40 respondents got a perfect score one the survey; furthermore, after running a Shapiro-Wilk test for normality on the scores, I fail to reject the null at the 5 percent level, with a p-value of 0.123. This means that there is not sufficient evidence to reject the claim that the data was drawn from a normal distribution.

Per the results of the survey, it seems like humans are not overwhelmingly able to tell the difference between human and computer generated video game sprite images. Furthermore, the results show that the images were almost indistinguishable, implying the computer generated images were able to simulate creativity.

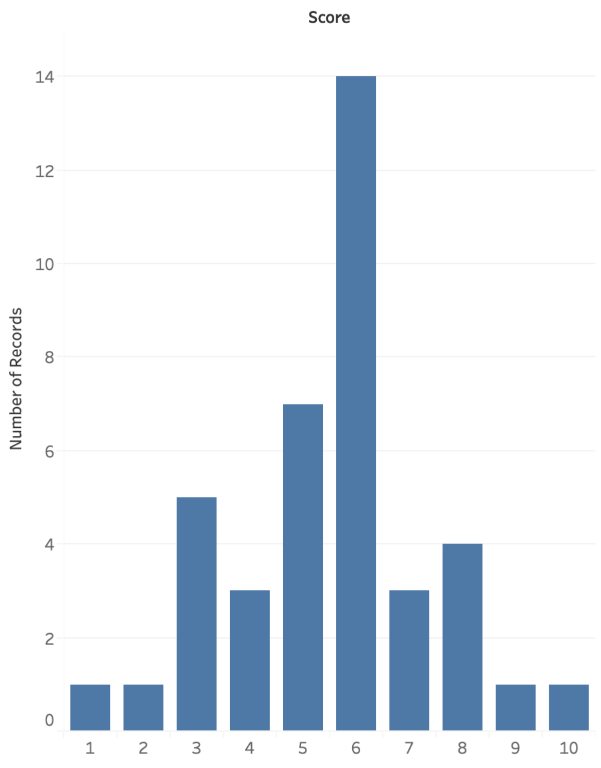


Figure 15: Survey Scores

V. Discussion

In this paper, I proposed a Generative Adversarial Network for the generation of video game sprite images using a novel dataset. The GAN was trained on 4,272 images categorized as characters and successfully generated images both mimicking the training images and creating novel video game sprites images with dimensions 32x32x3. In addition, the data cleaning method, which involved eliminating too similar images and cropping before resizing, resulted in better images.

The generated images were overwhelmingly humanoid in structure, with distinctive arms, legs, bodies, and heads. I believe the biggest flaw of the images is the lack of color consistency throughout entire body parts. Some generated sprites have randomly colored pixels in the middle of their bodies, which make them less likely to be classified as human generated.

This work could be useful for video game designers who want to make each character they interact with unique. If each character was unique, it would make for a completely unique gaming experience. If these characters were also assigned generated statistics and backstories, the potential for in game activities greatly grows. In addition, the potential for boredom due to repetition is likely to decrease.

When considering the ethical implications of GANs, if we can create computer generated art that has no discernible creative difference between human and computer-generated art, there could potentially be an influx of novel art hitting the market. If consumers viewed these two sources of art as perfect substitutes, then according to the law of supply and demand, which states that there is an inverse relationship between supply and the price of a good, the price of art will decrease. This would add economic strain on artists who, for the most part, are not in a highly paid profession. In addition, the computer would amplify the inherent biases held within the art and could lead to the possible stagnation of creativity if the same input data is used.

There are multiple sources for future work stemming from this project. It would be interesting to creating novel video game levels, with generated levels and characters. Then, the training agent would be trained using reinforcement learning to create a computer generated and played game.

Another source for future work would be to add a similarity score to the loss function [5]. Then, as the Creative Adversarial Network punished on similarity of style, this model would punish on similarity of images to avoid sprites that are copies of the data.

Code and image generation are available at: `https://github.com/thomasg8/SpriteGAN`

[1] Opengameart.org.

[2] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units, 2016.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018.

[4] Christie's. Is artificial intelligence set to become art's next medium?: Christie's, Dec 2018.

[5] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks, 2016.

[6] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2016.

[7] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks, generating "art" by learning about styles and deviating from style norms, 2017.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-

Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017.

[10] Lewis Horsley and Diego Perez Liebana. Building an automatic sprite generator with deep convolutional generative adversarial networks. pages 134–141, 08 2017.

[11] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks, 2017.

[12] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[14] Johnson J. Yeung S. Li, F. Module 2: Convolutional neural networks, 2019.

[15] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study, 2017.

[16] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[17] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge?, 2018.

[18] Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. pages 189–204, 03 2017.

[19] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[20] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[21] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training, 2016.

[22] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net, 2014.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[24] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[25] Ruben Rodriguez Torrado, Ahmed Khalifa, Michael Cerny Green, Niels Justesen, Sebastian Risi, and Julian Togelius. Bootstrapping conditional gans for video game level generation, 2019.

[26] Shizhan Zhu, Sanja Fidler, Raquel Urtasun, Dahua Lin, and Chen Change Loy. Be your own prada: Fashion synthesis with structural coherence, 2017.