Generative Adversarial Network Ensemble

Cassidy Klement Nicholas McGinty Christian Picofazzi
Drexel University, Philadelphia, PA, USA
College of Computing and Informatics
{cck47, nm3249, cxp32}@drexel.edu

***Abstract*—Generative Adversarial Networks consist of two primary components: a Generator and a Discriminator. Generators synthesize random images which are then mixed in with ground truth images and forward propagate though the Discriminator. The Discriminator is essentially a binary classification model with the classes being real or fake image. As the Discriminator begins to learn to differentiate between real and fake, the Generator must continue to learn how to produce better fake images to fool the Discriminator. Our project takes a look at training GANs per channel in an image and explore the observations each GAN makes as it extracts different features from each channel.**

***Keywords*—** machine learning, generative adversarial networks, cupy, ensemble, ground truth

I. Introduction

A cartoon is a type of illustration in which the subjects are drawn in an exaggerated manner. Cartoons are often, but not always, animated. The first animated cartoon was created by Émile Cohl in 1908 [1]. This animation, *Fantasmagorie*, was a short file composed of 700 black-and-white, hand-drawn illustrations and gave way to the rise of short animated film popularity that came in 1910.

Over a decade later, cartoons and animated films are still in high demand by both children and adults alike. Cartoons allow the viewers to become engaged in a fictional, and often caricatural, representation of silly antics and serious situations surrounding characters. Besides being a staple in many people's youths and serving as a form of entertainment and leisure, cartoons have been found to have positive effects on children's behavioral development and learning. Cartoons have the ability to quickly capture and maintain the attention of children, and while naturally this could result in negative consequences such as addiction or aggressive behavior depending on the type and volume of content consumed, cartoons have been shown to increase language acquisition and socialization in children [2].

Regardless of the potential impacts, cartoons are recognized as a form of art and are enjoyed by people of all ages. With this in mind, the idea of being able to provide a means to not only generate custom cartoons given a set of desired attributes but also to generate art is a very intriguing one. With a wealth of prior works existing on training generative adversarial networks (GAN) on multi-channel images yielding impressive results, this project aims to explore an ensembled approach to GAN training (i.e. training a single GAN per channel).

II. Related Work

A GAN consists of two primary components: a discriminator and a generator [3]. The primary goal of the discriminator is to distinguish between a set of real data and a set of generated data and accurately classify the samples as real or fake. The generator's primary goal is to produce fake data that can reliably pass as real data when classified by the discriminator. Since the creation of GANs, many applications have arisen. From generating fake images to colorizing black and white images and producing higher resolution images [4], the possible use cases for GANs are unbounded.

Focusing on the application of generating images, GANs have been used in previous work to synthesize medical images in the way of CT and MRI scans, for example. This is a particularly interesting use case, since past work has noted that robust and substantial medical datasets are considerably difficult to cultivate, being as the desired data is sensitive and requires specific domain knowledge in order to interpret [5]. As a result, assembling these medical datasets is costly and time-consuming. Skandarania et al. [5] noted that GANs could be used to generate realistic images and generated synthetic medical images from the limited existing data in order to validate whether or not these generated images could be used in the place of real medical images. As part of their validation efforts, they performed a "visual Turing test" in which they presented a half and half mixture of real medical scans and images generated by their StyleGAN to medical experts with many years of experience reading medical images. The experts were only able to achieve a 60% accuracy, indicating the high level of realism GANs are capable of producing.

More closely related to this project, prior work has been conducted to generate anime images using GANs [6]. The authors presented the argument that fake anime images could be useful both to anime artists for inspiration when doing character design, as well as to video game artists looking to decrease design costs as the resulting images are copyright-free. Additionally, the authors noted that the images that were generated by their GAN were of considerably low resolution due to the low resolution of available training images. They employed the use of a super resolution GAN to drastically increase the resolution of the generated images.

While both image generating related works described maintain similarities with this project, both works utilize a single network to train on the RBG image datasets. This project explores the intentional separation of RGB images into multiple channels and training a network on each of the channels in order to investigate the realism of channels generated in isolation and then combined to produce a single image.

III. Dataset and Features

The data we used train our GANs was Google's Cartoon dataset[1]. The full dataset consists of two disassociated sets, a set containing 100,000 sample images and a set containing 10,000 sample images. For our project experimentation, we worked exclusively with the 10,000 sample dataset, leaving room for future expansion with training on either the larger dataset or even potentially combining both datasets.

Each image has dimensions of 500 by 500 and includes an associated 18 attribute csv (comma separated values) metadata file that defines various characteristics of the cartoon face such as *hair_color*, *face_shape*, *eye_lashes*, etc. The RGBA (Red-Green-Blue-Alpha) color model was used, giving each image 4 trainable channels for our GANs to learn.

Regarding the associated image attributes, Google notes that there are approximately 10^{13} possible attribute combinations. While this project showcases the separation of a few hand-picked target attributes, this high level of variation implies that there exists the potential for highly customized image generation, should our GANs

[1]https://google.github.io/cartoonset

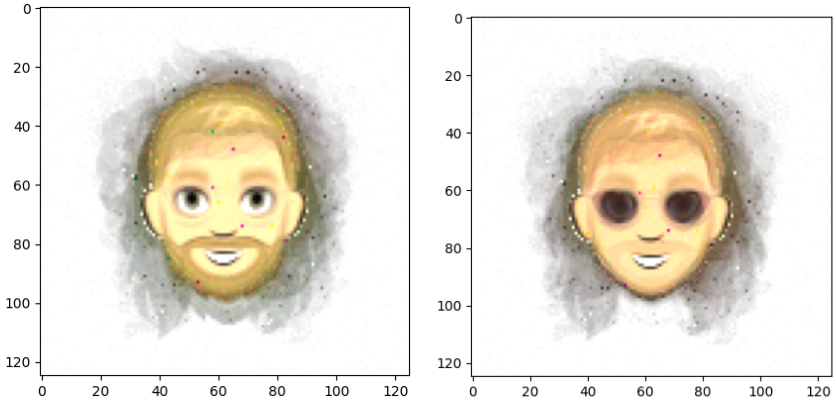


Figure 1: Generated Images from Facial Hair Subset and Shaded Glasses Subset, Respectively

be expanded to allow for the selection of any combination of the 18 attribute values.

```
"eye_angle", 1, 3
"eye_lashes", 0, 2
"eye_lid", 0, 2
"chin_length", 2, 3
"eyebrow_weight", 1, 2
"eyebrow_shape", 10, 14
"eyebrow_thickness", 3, 4
"face_shape", 1, 7
"facial_hair", 13, 15
"hair", 74, 111
"eye_color", 1, 5
"face_color", 3, 11
"hair_color", 8, 10
"glasses", 9, 12
"glasses_color", 6, 7
"eye_slant", 2, 3
"eyebrow_width", 1, 3
"eye_eyebrow_distance", 2, 3
```

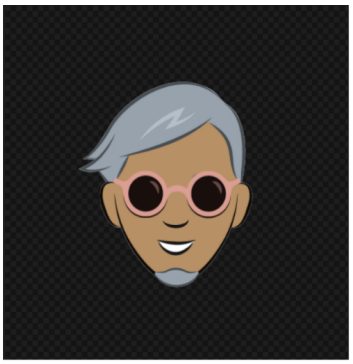


Figure 2: Associated attributes with Ground Truth Image

The graphic in Figure 2 depicts a ground truth image with its associated metadata csv file. The columns are defined from left to right as *attribute_name*, *attribute_value*, *total_unique_values*. Working with such a thoroughly annotated dataset made it very easy to parse different images into a subset to train on when we wanted to synthesize images with particular attributes.

IV. Methods

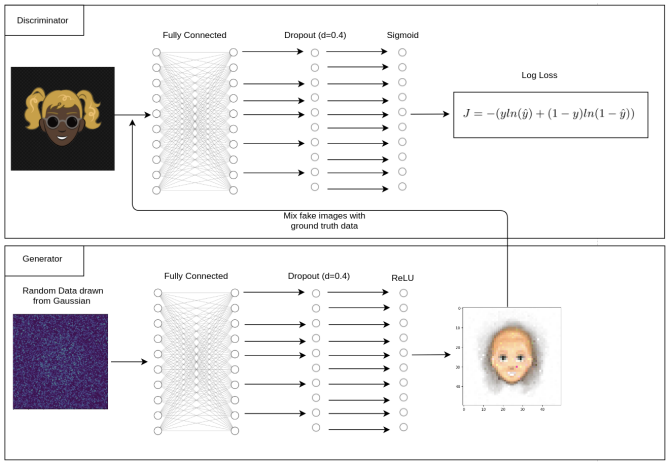


Figure 3: GAN architecture

As depicted in Figure 3, each of the 4 channels (RGBA) was trained on a separate GAN. Each GAN was implemented with the same architecture. The generator consisted of a single fully connected layer, a dropout layer, and a Rectified Linear Unit (ReLU) activation layer. The fully connected layer had D input nodes and D output nodes, with D representing the number of features. The input to the generator's fully connected layer was initialized from a Gaussian distribution of the real dataset and the weights were initialized as values between $\pm 10^{-6}$. The generator's dropout layer had a dropout rate of 0.4, meaning that 40% of the edges going to the ReLU activation layer from the fully connected layer were dropped (the corresponding node weights were set to 0). The output from the generator's dropout layer served as the input to the ReLU activation layer, where values less than 0 (negative values) were converted to 0. Since the output of the ReLU activation layer was treated as the fake data fed into the discriminator, negative values were dropped as all pixel values should be greater than or equal to 0.

The discriminator consisted of a single fully connected layer, a dropout layer, and a logistic (sigmoid) activation layer. The fully connected layer had D input nodes and 1 output node, to represent the single channel feature space. The input to the discriminator's fully connected layer was a single 2-dimensional image channel, flattened into a 1-dimensional vector, and the weights were also initialized as values between $\pm 10^{-6}$. The discriminator's dropout layer also had a dropout rate of 0.4.

The GANs were implemented using batch gradient descent, with a batch size of 50. This means that 50 samples were randomly selected from the real dataset and 50 fake samples were generated by the generator and combined to serve as the input to the discriminator's fully connected layer. The global learning rate was chosen to be 15^{-7}. Each GAN was trained for 1250 iterations.

As previously mentioned, the original images were sized at 500x500. With 4 channels (RGBA), the resulting feature space was 1 million cumulatively, or 250,000 per channel. We were unable to work with such a large feature space locally without running out of memory (perhaps by utilizing cloud technology which would put resource scaling in our control, we would be able to get around memory issues, but for the duration of this project, we were unable to use cloud resources). As a result, we experimented with decreasing the size of the feature space by scaling down the images. The largest image size we were successfully able to work with was 125x125.

An additional area of interest for this project was to be able to provide some level of control over the attributes that were used to train the model. Ideally, the selected attributes would be represented in the resulting generated images. Through manual inspection, training the model on the entire dataset resulted in images that represented a combination of all of the attribute variations. This can be seen

unmistakably with the generated image in Figure 6. The predominant cartoon face appears to be a person with light skin and short hair. However, we can see traces of different hair lengths and styles, as well as facial hair and rings surrounding the eyes which appear to be potential eye wear.

To explore how the active separation of certain attribute values in the training data would affect the resulting images, we hand-selected several attributes and supported values. For experimentation purposes, we chose to isolate two separate training sets: a set where all the samples had facial hair and a set where all the samples had shaded glasses. Training on each of the sets resulted in the images in Figure 1. As apparent through visual inspection, the facial hair and shaded glasses features are more predominant in the generated images trained on the training subsets than they were in the generated image trained on the full dataset.

A. Accelerated Computing: CuPy

Even with the reduction in image size from 500x500 to 125x125, the feature space for a single channel was still quite large (15,625 features). Considering that we desired to train for 1250 iterations, we wanted a method to speed up the training time for our GANs. Since our original GAN components were created using Python's NumPy[2] library to represent the vectorized images, we were able to easily use CuPy[3] as a drop-in replacement for the NumPy arrays.

CuPy is an open-source library that implements NumPy arrays on Nvidia GPUs by utilizing the CUDA platform and CUDA-related libraries. Integrating with CuPy allowed us to take advantage of the parallel computing platform and model provided by CUDA to speed up matrix computations using GPU-accelerated computing.

With the use of CuPy, we were successfully able to observe a fairly significant decrease in computation time. Using NumPy, the training time for a single channel (15,625 features) over 1250 iterations took approximately 35 minutes to complete. After we introduced CuPy into the code base, we observed an 80% decrease in training time. The training time for a single channel over 1250 iterations with CuPy took approximately 7 minutes to complete.

B. Greyscale

Greyscaling our images was the first pre-processing step we took when our GAN implementation was nascent. Greyscaled images only possess 1 channel of values. A channel is represented as a HxW (height by width) matrix of unsigned 8 bit integers (range of 0-255) that compose digital images. Traditionally in image processing, images are represented with a RGB (Red-Green-Blue) color model, where each color can be thought of as a channel.

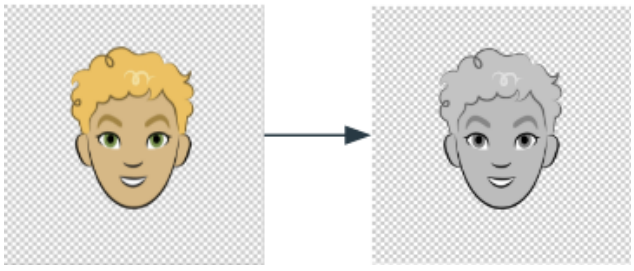


Figure 4: RGBA image to greyscale

In our case, the images from our dataset being from the RGBA color model possessed 4 channels, one for each of the color in the model (Red-Blue-Green-Alpha). Reducing from four channels to 1 greyscaled channel would reduce our image footprint size by 75%. Thus, greyscaling would facilitate faster model training which was our primary motivation for taking this approach while we were still defining our GAN architecture.

[2]https://numpy.org/

[3]https://cupy.dev/

C. Dropout: Issues with Overfitting

Initially when we designed the architecture for our GAN, we intended to keep it as simple as possible in an attempt to avoid any issues with overfitting. The original design included a single fully connected layer, a single activation layer, and a single objective function for the discriminator and the generator. With this architecture, we noticed that overfitting was still an issue, and the best images we could produce were markedly low fidelity (an example is given in figure 5).



Figure 5: Generated Image Without Dropout Layers

In an attempt to address the overfitting that we observed, we introduced a dropout layer into both the discriminator and the generator. Neural networks often exhibit overfitting, and a reliable way to get around overfitting is to have an ensemble of different model configurations. Of course, having multiple models means additional training time and maintenance expenses. To simulate the multi-model configuration approach, dropout regularization can be used, which randomly drops out nodes during the training process. Dropout regularization is computationally inexpensive and effective in improving a model's generalization.

Using a dropout rate of 0.4 for both the discriminator's and generator's dropout layers, we saw a significant improvement in generalization and our model yielded higher fidelity images, as show in Figure 6.

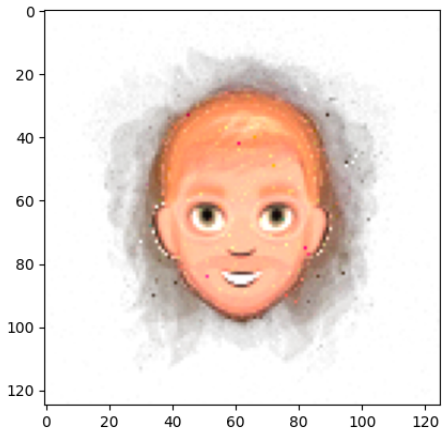


Figure 6: Generated Image With Dropout Layers

D. Ensemble GAN

Initially our interest in pursuing a multi-GAN or Ensemble GAN approach was to explore the learn-able traits across the RGBA color model. However this approach was necessary due to resource constraints. Our team used a single NVIDIA RTX 2080 Ti GPU that boasts 4352 CUDA Cores and 11 GB of GDDR6 VRAM. A sample

RGBA image from our dataset had dimensions of (500, 500, 4) or height, width and channels, respectively. To train our GAN we first attempted to reshape the image by flattening the first two dimensions to yield a new image representation with a dimension of (500*500, 4) or (250,000, 4). We then joined each 250,000 value channel into a single contiguous row vector consisting of 1 million values. Our primary data structure we used to store image data was CuPy arrays. Storing 1 million values for each image quickly consumed enough GPU VRAM to where we exceeded our memory limit. We pivoted to reducing our image set down to a moderate 125x125 (height, width) and separated each channel into its own row vector to train on individually.

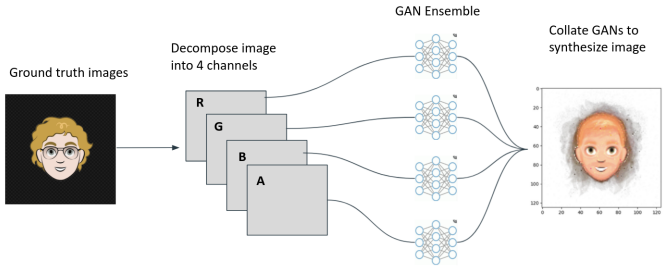


Figure 7: Generative Adversarial Network Ensemble Approach

In Figure 7, a ground truth image is reduced from its original size of 500x500 to 125x125 and then separated into 4 channels where each channel is used in a separate GAN for training. After the four GANs were completed with training, we took the synthesized channel image from each GAN and reshaped the 4 row vectors to our original reduced dimension of (125, 125, 4). This stacking of channels allowed us to create a four channel synthesized image. During our training we took images samples from each channel at 100 epochs which is depicted in Figure 8.

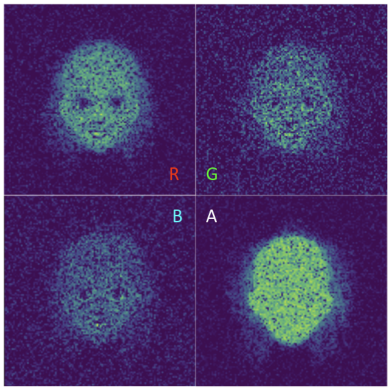


Figure 8: Synthesized channels after 100 epochs

V. Evaluation

In order to quantitatively measure the performance of our GANs, we measured the loss of both the discriminator and the generator. Success in our case would mean that the discriminator should experience a significant amount of loss, indicating that the generator was successfully able to deceive the discriminator. Similarly, we would expect the generator to have a low loss, indicating that the generated images would be classified as "real."

A. Discriminator Loss

Since the discriminator's objective function is the log loss objective function, $J_D = -(yln(\hat{y})) + (1-y)ln(1-\hat{y})$, and the actual and predicted values are a 50-50 split between ground truth images and generated images. As a result, we would anticipate a log loss convergence of around 0.5, indicating that the half of the target values that are generated by the generator were classified incorrectly by the discriminator.

Based off the results of the log loss function over time in Figure 9, we can see that the log loss for all 4 channels hovers between 0.45 and 0.5. This is a good indicator that the discriminator struggled to distinguish between the real and generated images.

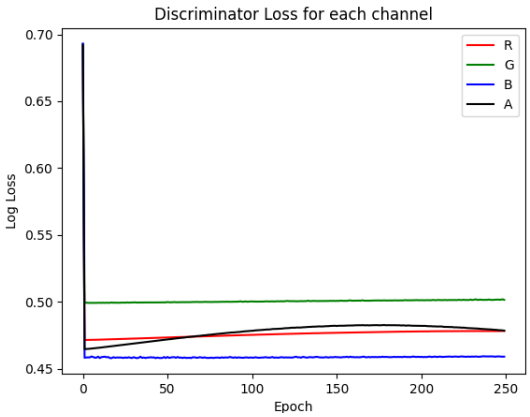


Figure 9: Discriminator Loss

B. Generator Loss

The objective function used in the discriminator is $J_G = -ln(\hat{y_f})$. The goal of this objective function is to measure whether or not the output yielded by the discriminator with respect to the generated samples is close to zero, indicating a zero loss from the discriminator's perspective. This perceived zero loss means that the generated images are yielding high probabilities that the generated samples are actually ground truth samples. A logarithmically decreasing loss over time, eventually converging towards 0, would indicate success.

Looking at the graph of the generator's loss function over time in Figure 10, we can see that the loss does decrease logarithmically. This indicates that the generator improves in generating images closer to the ground truth with each iteration. Although the graph shows the loss is above 0.5 as the lower bound, we can see that the loss per channel is declining and we would expect it to continue to approach 0 with every iteration.

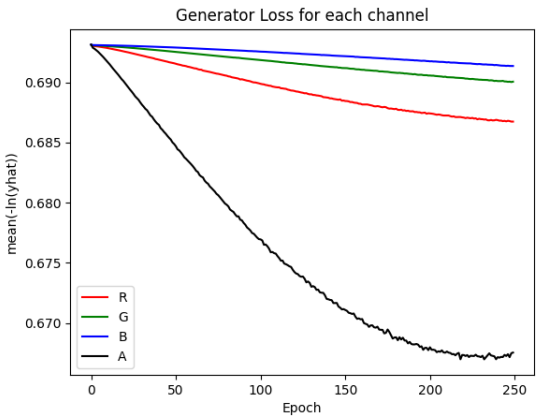


Figure 10: Generator Loss

C. Ethical ML

A qualitative review of our synthesized images raised some concerns about potential bias in our dataset. When controlling for no attributes, our GANs consistently generated images that are more representative of white males. This is a curious phenomena as there

was no attribute provided in the metadata that controlled for sex. There were however attributes pertaining to racial representation. This attribute, known as *face_color*, consisted of 11 values with labels from 0-10.

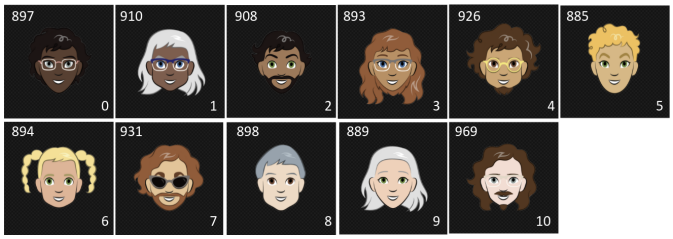


Figure 11: face_color attribute frequency and class label

In Figure 11, you can see one observation from each of the *face_color* classes with the label of the image class in the bottom right, along with its frequency in our dataset in the top left. A simple observation suggests the dataset is somewhat uniformly distributed across the 11 labels, with about 900 observations in each attribute class. For this semi-uniform distribution to synthesise such biased output suggests there is some additional work to do when analyzing the attributes of the dataset. Approaching this investigation by leveraging explainable or interpretable machine learning techniques would be work that we would like to pursue in the future.

VI. Conclusion

In summary, we explored the use of an ensembled GAN approach where we trained a separate GAN on each channel of RGBA images. We maintained a fairly simple network architecture and employed the use of dropout regularization to remediate overfitting that was observed. We experimented with a simple proof-of-concept on attribute selection to train GANs to generate images with a desired set of features as opposed to solely training the GANs on the full dataset.

In order to decrease computation time, we integrated with CuPy to take advantage of GPU-accelerated computing and observed drastic time decreases, which allowed us to conduct more thorough experimentation without being severely crippled by training times typical with working with large datasets and/or feature spaces.

Overall, we were satisfied with the images that were generated by our ensembled GAN method. Starting out with this project, we were uncertain whether or not isolating the channels would yield good results. In addition to the evaluation results, it is visually observable to the human eye that the generated images are sensible with respect to the cartoon training samples and there is no doubt that the ensembled GAN approach was able to successfully learn how to generate a cartoon face.

VII. Future Work

A. Convolutional Layers

Our team managed to get a working convolutional and max pooling layer class implemented but due to time constraints, we were unable to integrate this into our GAN's architecture. The convolutional layer was able to take in a known filter (i.e. edge detector layer) and output a convolution of that layer with a ReLU and max pooling applied.



Figure 12: Vertical Filter



Figure 13: Horizontal Filter

With more time, our team would like to be able to implement back propagation in the convolutional layers so that the layer may be able to learn unique components of our data. We would also like to work with higher dimensional images within the cartoon set, convolve down to some trainable feature map space and train our networks. After training is complete, we would add a post-processing step to return back to the original dimensionality in the hopes of producing higher fidelity images. This would hopefully help us produce a Deep Convolutional Generative Adversarial Network (DCGAN), which is just a series of convolutions that convert random noise into the dimension of an image before it is passed to the generator of a GAN. See Figure 14.

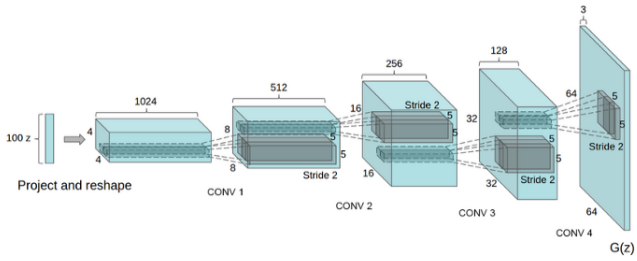


Figure 14: Deep Convolutional Generative Adversarial Network [7]

B. Deep Regret Analytic Generative Adversarial Network

A way to improve training time, stability and modeling performance is with a novel gradient penalty scheme called DRAGAN (Deep Regret Analytic Generative Adversarial Network), which can be used across a variety of architectures and objective functions. [8].

C. Explainable AI

Our synthesized images appeared to be somewhat biased in their representation of the *face_color* attribute. This could be a mathematical phenomenon, but our team would like to do more investigation in the fairness of our dataset and explore any potential explanations as to why the GAN extracted the features it did during training.

D. Attribute-Driven Image Synthesis

As of right now, our architecture is limited to producing images with particular features only when we control for a given attribute. For example, consider N to be the number of unique values for a given attribute, let that attribute be *hair* where N is 110. For us to accurately synthesize images for each value of the *hair* attribute, we would be required to to separate our dataset into N subsets and train N GANs. This architecture does not scale well due to the high number of attributes (18) and their possible values. Exploring an architecture or method that could provide more versatility in generated output by consuming a subset of attributes and producing images with those features would be a worthwhile investigation.

References

[1] "Fantasmagorie, the world's first fully animated cartoon, was released on aug 17, 1908," https://www.indiatoday.in/education-today/gk-current-affairs/story/fantasmagorie-first-animated-film-1030219-2017-08-18.

[2] S. A. Ghilzai1, R. Alam, Z. Ahmad, A. Shaukat, and S. S. Noor, "Impact of cartoon programs on children's language and behavior," *Insights in Language Society and Culture*, 2017.
[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *arXiv*, 2014.
[4] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv*, 2017.
[5] Y. Skandarania, P.-M. Jodoinc, and A. Lalandea, "Gans for medical image synthesis: An empirical study," *arXiv*, 2021.
[6] Y. J. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the automatic anime characters creation with generative adversarial networks," *arXiv*, 2017.
[7] A. Radford and L. Metz, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv*, 2016.
[8] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of gans," *arXiv*, 2017.