A Multiple-GAN Generation of Images from NASA Hubble Space Telescope

Sameerah Helal **Radhika Kulkarni** **Katherine Shaw**

Contributions

Sameerah Helal

- Tested implementation of several nonfunctional GANs from different sources
- Successfully used scraping tool from GitHub source to scrape images from Flickr
- Uploaded all .py files and image folders to Alan Server, set up initial directories
- Debugged and modified WGAN, DCGAN, and GAN code to be functional on local machines and servers
- Modified all scripts to accept custom image sources, additional argparse arguments, and save full scale images after program is run
- Curated image subsets to
- Modified script and tuned parameters extensively for dcgan.py
- Wrote Final Report - Implementation and DCGAN Results

Radhika Kulkarni

- Identified, filtered images of NASA Hubble Space Telescope on Flickr for photos and large images, then webscraped from Flickr
- Identified most effective GAN implementations after multiple tests of ineffective methods
- Set up /scratch directory, transferred all .py files and existing image folders in Alan server
- Modified script and implemented, tuned algorithm in wgan.py
- Transferred output from remote server to local, set up and used submit function in email
- Organized meetings and planning, identified proper implementation PyTorch-GAN
- Implemented modified wgan.py on subsetted image categories with tuning
- Wrote part of proposal; Final Report - Methodology and WGAN Results

Katherine Shaw

- Researched multiple nonfunctional and functional GANs from various sources thoroughly
- Webscraped 713 images from Flickr
- Modified script through Vim, implemented algorithm, and tuned parameters extensively in gan.py
- Debugged code for errors
- Set up command to transfer output from remote server to local, used submit function
- Implemented modified gan.py on subsetted image categories with tuning
- Wrote up and formatted proposal; Final Report - Introduction and GAN Results

I. Introduction

Generative models in machine learning are often applied in the context of images. Generative adversarial networks (GANs) have become a popular algorithm used to train these models in an unsupervised manner that involves supervised steps. The main idea is to extract and learn about the patterns or regularities present in input data in order to generate new examples that are similar to the original dataset.

We used three General Adversarial Networks, or GANs: GAN, WGAN, and DCGAN, to generate brand new galaxy images from an original dataset of 713 images from the NASA Hubble Space Telescope and sourced from Flickr. Our problem was to tune these images using GAN, WGAN, and DCGAN to determine which algorithm outputs images with more diversity and higher quality. Furthermore, we wanted to identify the best tuning parameters for each algorithm in order to maximize their potential of producing optimal images. Some of the parameters we will experiment with include number of epochs, batch size, and learning rate; as well as other, model-specific parameters.

Although space exploration is continuously developing and advancing, there will always be infinitely many galaxies and realms we have yet to witness. Our first motivation was to visualize these unexplored realms in space by producing one-of-a-kind images of domains that could potentially exist. Our second motivation was to create unique space wallpapers for phone users who are looking for original, distinctive, and custom-generated backgrounds. We also hoped to convince viewers that the generated images are actual images of space. Our hypothesis, or priori, was that WGAN and DCGAN would produce better and more realistic images of a higher quality than regular GAN.

Ultimately, we found that our hypothesis was true in some aspects. Our results show that WGAN and DCGAN are better than GAN in certain facets; for example, WGAN has a faster runtime than DCGAN and GAN while DCGAN generates more vivid and colorful images. However, GAN produces higher quality images with more variation for this particular original dataset. Perhaps on other datasets with different features or characteristics, our hypothesis may have been proven true. A universal result we found was that tuning parameters for all three algorithms improved the results substantially; every dataset has unique parameters that need to be tuned for optimal performance.

II. Methodology

A General Adversarial Network can be thought of as a game that involves two players as differentiable functions, a discriminator D and a generator G, in which G's goal is to fool D that whatever G has generated comes from the training data's distribution. G and D are deep neural networks. G takes noise z, a latent variable taken from a prior distribution, as input. This $G(z)$ is actually x samples taken from the *pmodel* distribution, and is evaluated by D to determine, through supervised learning methods, if the input has come from the same distribution as the training data or not. G's objective is to convince D that the input has come from the training data, while D's objective is to prove that G's data is fake. If D finds that G's data is fake, G tries again and again to convince it until D is fooled, the game ends and thus the generated data is close to the actual. This is a Nash Equilibrium: the cost functions for D's and G's associated parameters are minimized.

The dataset of 713 NASA Hubble Space Telescope images that we used for this project was obtained from Flickr through web scraping, by permission of the Flickr API. Most of the background of these images were black or contained the heavenly bodies, a riot of amalgamous color. Note that the original images taken by Hubble are not in actuality colored, but were taken in black and white and subsequently underwent image colorization to enhance the details in these celestial forms. These original images were of very high quality and were similar enough to each other that the GAN algorithms were able to find a stable structure to train on; namely, a solid white circular center with varying shades of rings around it, and a black background. As the hyperparameters were tuned, we aimed to generate more structural variety and colors as one might find in space, and for more interesting wallpapers.

We decided to use multiple GAN algorithms to generate our images because we were curious about the changes that GANs developed after 2016 would make on our largely similar images of space. We also hoped that there would be marked improvements in image quality, time and attention to detail. The first GAN that we used was the original GAN algorithm; we then chose WGAN, short for Wasserstein GAN, because of its highlights of a more stable learning process and its claim that it is more resistant to mode collapse and vanishing gradient. Despite the fact that the problems that

WGAN claims to solve are inherent in DCGAN, the latter is generally regarded as faster than the other GANs we looked at, so we have a trade-off between time and resistance to common image generation issues.

The image datasets that we trained and tested on for these three GAN algorithms were of sizes 10, 100, 300, and 700 images. Additional subsets include those with heavenly bodies featuring rings, swirl patterns, or stars.

III. Implementation Details

As Generative Adversarial Networks are time consuming in construction, training, and tuning, we chose to use existing networks to train on. After attempting several iterations of GAN from different sources, we decided on a PyTorch implementation by Erik Linder-Norén[1]. Out of his dozens of varieties, we selected WGAN, DCGAN, and regular GAN.

Using an API key [2], we scraped approximately 700 images from the Flickr search query "galaxy". Once these were vetted and uploaded to the Alan server provided by the UC Davis Statistics Department, we began modifying the GAN Python scripts for compatibility with our problem. Our alterations included allowing the networks to train on custom datasets, saving images in our desired format, and various other adjustments for model tuning purposes.

Before tuning parameters, we used the default model parameters with varying image sizes and quantities. Starting with 10 images, which resulted in poor outputs, we increased the number to 100, 300, and eventually the full 700. We also experimented with curating different, less heterogenous subsets of the dataset: for example, a subset containing images of only stars, or only ring-like structures. Since our training images varied in size from 512 × 512 up to 1024 × 1024 dimensions, we tested up to that size.

Tuning the models involved changing many of the model-specific parameters, the first of which we tuned were epoch and batch size. Increasing the number of epochs for which the model trained from the default of 200 generally raised the quality of the generated images as well as the runtime. To varying results, we altered batch size, or the number of images trained on in each epoch, from the commonly used default of 64; as well as the learning rate, the regulator of the step size at each iteration with respect to the loss gradient as it moves toward the loss function minimum.

DCGAN and GAN use the Adam optimizer, which depends on the decay of first and second order momentum, or β_1 and β_2, respectively. Several resources suggested that the best values for these were 0.9 and 0.999, from which we deviated slightly, ultimately ending up deciding to keep the defaults of 0.5 and 0.999. A value that we did change considerably was the clip value for WGAN, which informs a measure called Wasserstein distance by clipping or limiting weights for its Lipschitz constraint. Clip value weights enforce the Lipschitz constraint $|f(x_1) - f(x_2)| \leq \mathbf{K} |x_1 - x_2|$ on the Wasserstein distance. Changing the clip value is tricky because if one increases it too much, it takes much longer for weights to "reach their limit"[3], or if one decreases it, it can cause vanishing gradients. Imposing this constraint is the biggest issue with WGAN, and, depending on the dataset, it may not create the desired high quality images.

IV. Results and Interpretation

We trained GAN, WGAN, and DCGAN models on the same Galaxy dataset, building up from 10 images to 100, 300, and finally 700. Our initial image size was the default of 28 × 28 (× 3 for the 3 color channels) pixels. We worked our way up to 128 × 128 (× 3), and finally to 512 × 512 (× 3) pixels. Recall that we hypothesized that WGAN and DCGAN would produce better images than GAN. This hypothesis turned out to be true in some ways, and not in others: WGAN trained on the images in much faster time than GAN, and DCGAN drew out far more color than GAN. Ultimately, however, the detail for regular GAN for this specific large scale dataset was superior.

GAN: Figure 1 is the result of training the regular GAN model with the default parameters and dataset. The results show mode collapse (also known as homogeneity or lack of variety) and static. Figure 2 is the result after tuning parameters

[1] https://github.com/eriklindernoren/PyTorch-GAN
[2] https://github.com/antiboredom/flickr-scrape
[3] https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

repeatedly, and was run with the values that were determined the best. Some of the parameters we changed and tested were number of channels, epochs, batch size, learning rate, sample interval, and the image dataset. Increasing the number of epochs consistently improved image quality and variation, but only minimally when we hit 500 epochs; we decided 400 was optimal in order to maximize quality and runtime. Increasing the number of images trained on also increased quality and diversity significantly. The best results used the default parameters of batch size (64, which achieved 98% accuracy) and learning rate (0.0002). Reducing batch size to 32 did not improve the results, and increasing the batch lowered the quality of the model. 0.0001, 0.001, and 0.01 are all common learning rates, but 0.001 produced unclear images and no variation (not worth the shorter runtime), while 0.0001 remedied some of the mode collapse but trained slowly and did not improve quality. 0.0002 seemed to be the sweet spot, despite appearing small compared to standard learning rates. 0.9 is a common value of β_1—a parameter that regulates the first moment decay of the ADAM optimization algorithm—but we used a value of 0.5 because it stabilized training. β_2 was 0.999 for all the resources we looked at. Figure 3 shows results after using the image subset "swirly" with specific parameters (see page 1 of the supplemental materials).

Figure 1. Run with 100 images, default: image size 28, 200 epochs

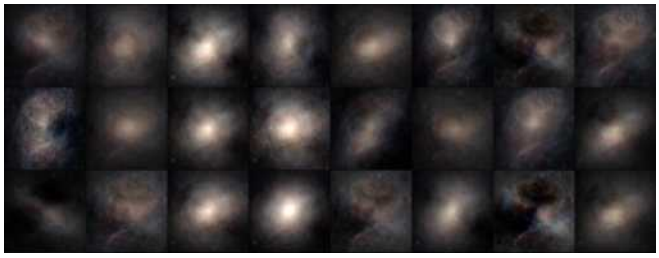

Figure 2. Run with 700 images, image size 512, 400 epochs

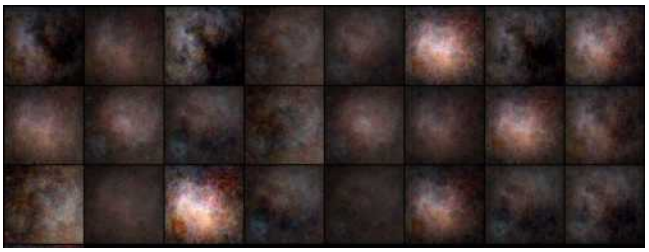

Figure 3. Run with subsetted images, image size 512, 400 epochs

WGAN: We trained WGAN on images of sizes starting with 10, 100, 300 and then 700 using various parameters, the most important of which was clip value weights. Figure 4 shows the base result using default parameters: 200 epochs, batch size of 64, learning rate of 0.0005, 5 critical steps, and clip value of 0.01 for 700 images. After weeks of training the model on different parameters, Figure 5 is the final result after adjusting clip value to be 0.2, image size to 512 × 512 dimensions for the whole dataset of over 700 images. After increasing the clip value, we began seeing different structures than the mode collapse. For this dataset, WGAN tended in most cases towards mode collapse, and as the researchers who invented the GAN admitted, even clipping the weights may not actually ensure the variety in results without normalizing batch sizes. Changing learning rate, batch size, number of critical steps and number of epochs had no visibly significant change for WGAN image quality and galaxy structures, except in image subsets such as "swirly", "ring-like", and "starry" (see supplemental material page 2 for more images). The best results for the 700 images of size 512 occurred with the default learning rate of 0.00005, epochs of 200, and batch size of 64; and with an altered clip value of 0.2.

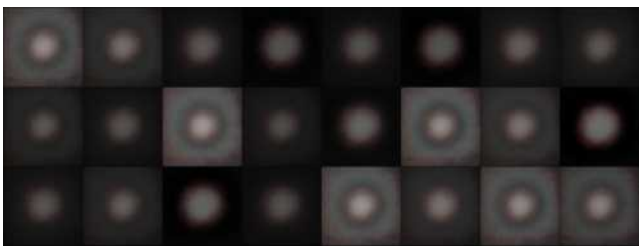

Figure 4. Run with default, clip-value 0.001, 700 images, image size 512

Figure 5. Run with default, clip value 0.2, 700 images, image size 512

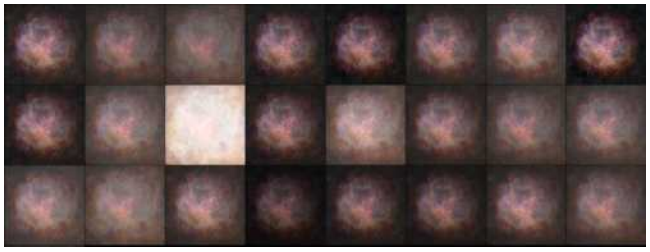

Figure 6. Run with default and learning rate of 0.0002, high quality swirl images, mode collapse persists w different parameters

DCGAN: We first trained DCGAN on the default parameters of 200 epochs and learning rate 0.002 with the relatively low number of images of 10; the result is shown in Figure 7. We then increased the number of images several times, up to 700; as well as increased the number of epochs to the end of generating more varied and defined images. However, this method failed multiple times, both in result and in output. To combat this, we decided to use slightly more homogeneous, curated rather than random subsets. After experimenting with increasing epochs, reducing batch size, and changing learning rate and β (both 1 and 2) we eventually arrived at 400 epochs (twice the default), 32 batch size (half the default), and 0.0002 learning rate (a tenth of the default) as parameters that resulted in reasonably good images. After much research we decided to keep β_1 and β_2 at their respective original values of 0.5 and 0.999. DCGAN had the unique advantage of producing, even with little tuning, and despite not being as finely detailed; results with more color than generated by other GANs. This is apparent in one of our final tuned images, trained on images of ring-like structures, in Figure 9. See supplemental materials page 3 for more of these results.

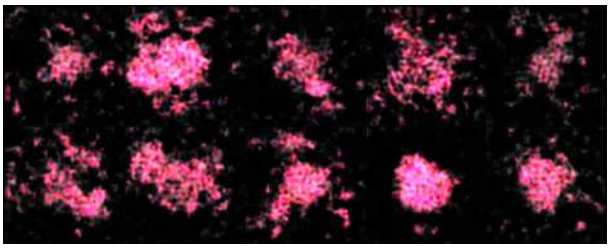

Figure 7. Run with default parameters and 10 random images.

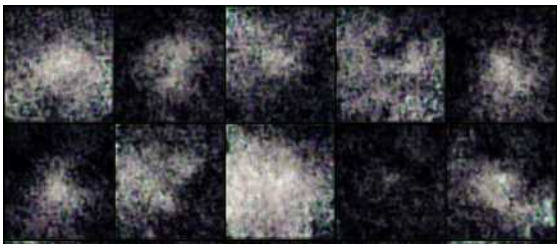

Figure 8. Run with 100 images, and parameters tuned to default.

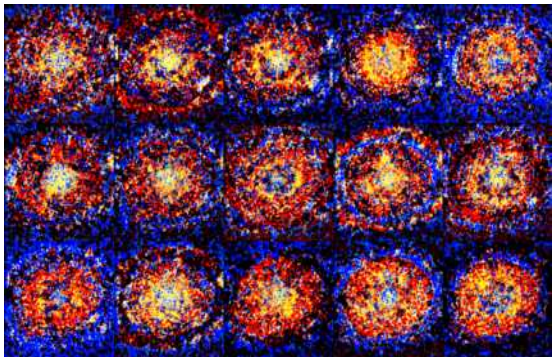

Figure 9. Run with subset, image size 512, learning rate 0.0002

From this project, we learned that it takes a lot of background mathematical knowledge to truly train the models by choosing the correct parameters in an efficient and targeted way. Manually changing and experimenting taught us patience and amazed us that we could perform such image generation with just these three GANs. There are so many more and newer GANs out there that require more time and advanced technology; in the future, we look forward to testing them out and seeing what images we could produce. With hyperparameter tuning and more powerful computers, the possibilities are endless. We enjoyed the results of our experiments and think that some of them, particularly the image subsets, could make GAN-tastic wallpapers.

V. Supplemental Materials