Generative Adversarial Networks: Assessment of Transfer Learning on Small Datasets

Hume Center for National Security and Technology

ESHAEP

Brian Lee

Abstract

The main objective of this project was to explore the capabilities of today's state-of-the-art generative adversarial networks (GANs) in image generation using transfer learning on limited datasets assimilated from Instagram. As a step in this process, a pipeline of tools was assembled to intake and filter image datasets compatible with GAN training requirements. Transfer learning is a technique which leverages a network pre-trained on a different, often larger, dataset. Transfer learning provides benefits in training time, computing requirements, and small dataset compatibility. It was found that pre-trained networks have noteworthy potential in adapting to new domains of data, even when using small datasets with moderate variability.

1. Introduction

Generative modeling is the unsupervised learning task of modeling the distribution of input training data so that sampling from the learned model results in examples that plausibly resemble examples from that real distribution [8]. Given this task, generative adversarial networks (GANs) offer an innovative solution. The basic GAN architecture in Figure 1 consists of two networks pitted against each other as adversaries: a generator network and a discriminator network. Note that in this standard architecture, random noise is fed into the generator to generate data from. The generator is tasked with learning the distribution of a real dataset to generate new examples mimicking the true data. The discriminator network is tasked with learning how to accurately classify the real and generated examples, which it notifies the generator as feedback [3].

Together, these networks and their loss functions are competing in what is known as a two-player min-

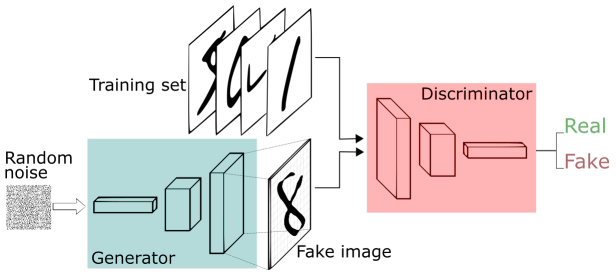

Figure 1. Basic generative adversarial network structure [9]

imax game. While the discriminator learns to maximize its classification successes, the generator learns to make examples that minimize the discriminator's classification successes. First, the distribution from which the input random noise is created is defined with a prior $p_z(z)$. Next, the nonlinear function that maps input to the data space is denoted $G(z)$. The resulting distribution of generated data is defined as p_g, which is being learned to resemble the distribution of the train-

ing data, defined as p_{data}. Lastly, the probability that a data point is from the training dataset rather than the generated one is denoted $D(x)$. Then, the minimax loss equation is defined with the value function $V(G, D)$, as described in the derivation by Goodfellow [3].

$$\min_G \max_D V(G, D) = \quad (1)$$

Currently, image data is the most popular domain of data in which GANs are trained []. Not only were the first GAN experiments trained with image data, but their most well-known and exciting applications typically involve some form of image synthesis. Among such foci include image generation for datasets [3], photo-realistic human face generation [5], and text-to-image generation [7].

However, with this popularity comes a point of concern. In many published works, generative adversarial networks are trained from scratch. Considering that image files are relatively large, training on large datasets over extended periods of time can prove incredibly costly in both time and computation [5]. One approach that mitigates these issues is performing transfer learning on an existing model. Transfer learning is the technique of applying a model that has previously been trained on one task to another task, domain, or subject. In this way, the model has already learned from a particular dataset common features of image data such as edges, curves, coloration, lighting behavior, among more, which allows it to adapt to a new domain more efficiently. Consequently, a critical component of this project is building tools to assemble datasets for training.

The primary objective of this project was to evaluate how quick and effective the transfer learning process is as well as its limitations depending on the training datasets we created. Additionally, we wanted to assess the viability of the datasets scraped from Instagram and assimilated. This particularly important for exploring domains of data limited by the specificity or abundance of appropriate examples.

Leveraging transfer learning in the training process is a sensible approach to address both objectives. Most significant, constraints on time and computational resources necessitated taking advantage of transfer learning. Transfer learning requires a fraction of iterations as does training from scratch, which allowed us to perform multiple runs and explore a wider space in. Additionally training GANs from scratch with the small datasets have been known to result in failure modes [2]. The chances of these occurrences can be minimized by using a pre-trained network and transfer learning. As results are obtained, we can explore what qualities of a dataset make it more suitable to use in training.

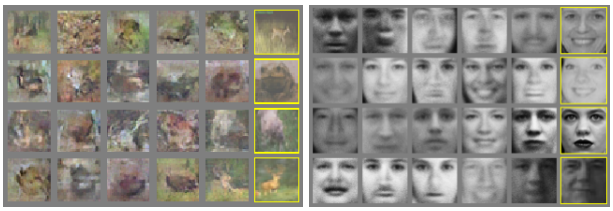

Figure 2. Left to right: Images generated by generative adversarial networks from Goodfellow et al. training on CIFAR-10 and TFD datasets [3]

2. Related Work

2.1. GANs (Goodfellow et al. 2014)

The generative adversarial network framework is a recent architecture proposed by Ian Goodfellow in 2014. Most existing approaches to generative modeling in the machine learning community at the time consisted of single-network architectures, such as deep belief networks or stacked convolutional autoencoders [3]. It is in this work that the two network architecture is proposed.

Goodfellow and colleagues demonstrate the viability of GANs in image generation through experiments on datasets such as MNIST, CIFAR-10 and the Toronto Face Database. In these results of this early stage of GANs, the generator network is produces images that are representative of the real examples, as shown in Figure 2. It was concluded that although it is unclear whether GANs generate better samples than those of current methods, they are ”at least competitive” with them and ”highlight the potential of the adversarial framework.” The publication puts the two-network format of GANs on a mainstream stage, and it has since been thought of as a state-of-the-art approach generative modeling using machine learning [3].

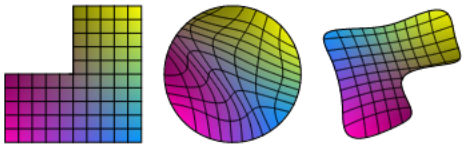

Figure 3. An illustraiton of the effects of the mapping network [6]. Note that the distribution in c) reflects that of the original in a) much more familiarly.

2.2. styleGAN (Karras et al. 2019)

Although many variations of the original GAN architecture have produced successful results, a persisting challenge with all architectures was a lack of control along the image synthesis process [6].

To address this challenge, researchers from NVIDIA reworked the generator's fundamental structure with a state-of-the-art architecture in StyleGAN. Traditional generators take initial latent code defined by a random distribution in the input layer only. In contrast, styleGAN takes advantage of a standalone mapping network that maps the random latent vectors from (Z) to an intermediate latent space, (W). Simply put, this mapping operation makes the features of the training dataset easier for the generator to learn. Consider that, in many datasets, the distribution of its features are often non-uniform. Consequently, random inputs from (Z), underrepresent more common features while overrepresenting ones that are less prominent in the data. Figure [] visualizes the benefits by the changes that are made to (Z) [6].

Learned affine transformations are applied to the vectors in (W) to specialize them into various styles, as is referenced by the name "styleGAN." These styles are then used "to control adaptive instance normalization" (AdaIN) operations after each convolution operation in the generator network. As the image is upsampled, various styles can be chosen and applied to the image accordingly. As a result, the generator is capable of producing photorealistic images with fine features that can be tuned and adjusted. Furthermore, styles can be mixed across images, allowing for a remarkable variation in generated examples [6].

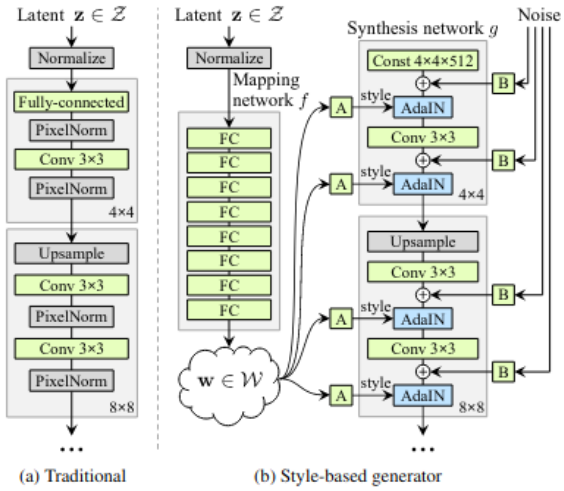

Figure 4. Traditional generator structure compared to that of styleGAN [6]. Note the use of a mapping network to create an intermediate latent space.

3. Materials and Methods

3.1. Data Pipeline

One early, yet important focus in the experiments was on the creation of the small-scale datasets used in the transfer learning process. In devising the various ways that such datasets could be developed, scraping public images from the social media site Instagram was a method of interest. Transfer learning experiments involve training the networks on small datasets of images different from those on which they were previously trained. Therefore, possessing both versatility and organization in finding image subjects was important. Instagram, an image-based platform that has hashtag functionality and a large volume of public images, allows for a simple approach to which collect and categorize data.

The Python package Instaloader was a crucial tool used to scrape image data. Developed by Alexander Graf and Andre Koch-Kramer (aandergr and Thammus on GitHub), Instaloader uses powerful functionality to download images and videos along with their metadata from Instagram posts in a streamlined manner. Naturally, the Python package Requests is heavily involved in this process. Data was scraped from each hashtag with a target of 1,000 images for raw image data. Due to multi-image posts, 1,500 to 2,000 images ended up being gathered per hashtag.

Figure 5. Examples of images scraped by Instaloader, after being given the hashtag "#latteart"

Figure 6. Left to right, top to bottom: Images of #fighterjet that would be rejected for low resolution, blurriness, grayscale, prominent text, human faces. The generator can have considerable trouble when attempting to generate the latter two elements of images in particular.

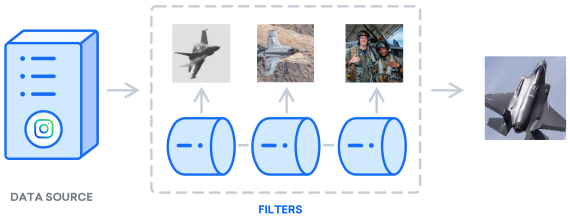

Figure 7. A Diagram of the current working data filtration pipeline. Various filters catch input images for criteria of unacceptable thresholds, leaving behind only usable images.

Challenges would arise during the scraping process. On the most basic level, Instaloader's built-in Requests rate-controller class would temporarily lock the accounts from downloading post media to avoid being flagged by Instagram automation detection. This slowed down data collection for extended periods of time. In addition, hashtags themselves have a degree of imprecision. On one hand, Instagram users will tag a post with a flurry of hashtags regardless of how relevant they are to the media in the post. As long as a hashtag being searched for is in the caption, Instaloader downloads it anyway. On the other hand, the media in a multi-image post gets downloaded, even if off-topic to the hashtag. In either case, these made the collected data of some hashtags far too varied and ambiguous for training purposes, where using a dataset of similar images is crucial. To remedy the latter issue, an intentional, cursory examination of hashtags was to be performed prior to selecting it and scraping as an appropriate subject.

Training networks with image data requires a standardized dataset of suitable images. The GAN must be trained on images of sufficient quality or images without elements that the generator will have significant trouble reproducing [1]. For example, if the dataset includes blurry images, blurry images will be generated. A data filtration pipeline was devised to iterate through all collected images and discard those that did not pass various thresholds of certain target criteria. Figure ??? contains examples of images that would be rejected by these filters. Simply put, each criterion acts as a filter that the image has to pass through on the path to standardization and dataset assembly. These filters included:

- Valid image file format (.jpg, .png, .webp, etc...)
- Pixel height and widths are both within at least 80% of a target resolution (i.e., 512×512)
- Sufficiently sharp images
- No grayscale images
- No significant or prominent bodies of text
- No visible human faces

Upon passing all filters, images are cropped and resized to a specified square size and then saved as new training examples. Numerous packages are used to process the image both in the filtering and resize operations. Pillow and OpenCV are two general-purpose Python modules used to read, edit, and save image data. For the facial recognition filter, GitHub user

Figure 8. Parameters need to be tuned when false positive and false negative classifications pass through the filter. Green boxes overlay all instances of text detection, true or false.

ageitgey's Python module provides functionality to detect presence, location, and recurrence of human faces. For the text recognition filter, OpenCV's Frozen Efficient and Accurate Scene Text (EAST) Detection deep learning model provided creative ways to identify text (Rosebrock). The package also provides functionality to draw boxes over the classified text, which was exceptionally helpful for threshold tuning and debugging.

The main challenge that continuously arose was manually tuning the actual thresholds of the filters. The pipeline, at various filters, would either reject images that are indeed good training examples or accept images that are poor training examples. In other words, it would filter too harshly at some checkpoints while filter too leniently at others. Running tests on toy datasets, outputs would have to be examined to tune parameters. For example, the Frozen East Text Detection network that was used for the text checkpoint would, at times, falsely classify parts of the image as text, when in actuality, the image simply contains dark edges against a light background. Much personal discernment needed to be done on a variety of classifications to find the right parameter. This applies to the other filters as well, although to a lesser extent.

3.2. Training

Training experiments were conducted on a computing cluster from Advanced Research Computing (ARC) at Virginia Tech. Remote connection to the cluster provided access to three GPU nodes with eight GPUs each, totalling to 24 maximum GPUs available to train at any given time. Training performance was measured using the Fréchet Inception Distance (FID) score, a metric measuring the similarity between two datasets. A lower FID score reflects greater resemblance of a dataset [4]. Python implementations for dataset formation, training, and generation are provided by NVIDIA's repository for StyleGAN3. Pretrained styleGAN3 are obtained from NVIDIA.

An initial round of experiments were run from NVIDIA's 512x512 AFHQ GAN network, pre-trained on animal face images, now onto a dataset of beach sunsets, scraped from "#beachsunsets" on Instagram. For these initial runs, we tried various permutations to the default run's parameters. Learning rates were generally adjusted such that the discriminator and generator learning rates matched eachother. Nudging them a few steps in either direction seemed to suggest that a moderate decrease in the learning rates yield marginal reductions in FID score. Most runs reached an oscillatory state around the mark of 3000 iterations, so further jobs were run for 3500 kimg.

For a second round of experiments, we explored more impactful parameters. Now holding a learning rate of 0.0016 constant, we passed various values for the "gamma" parameter. This gamma value is synonymous to the R1 regularization term and it is thought to be the most impactful parameter for training. Put simply, in training GANs, this R1 regularization term penalizes the discriminator from making too big of changes in its loss function in an effort to correct the discriminator stabilize training. Harsher values from the default gamma of 10 worked well, with 15 and 20 resulting in competitive FID scores. Deviations beyond 30 tended to raise the FID score from too much penalization on the discriminator. Additionally, we applied different augmentation techniques to the training. Augmentation is a mode of generating slight variations in the training dataset so that there are more examples from which to train the networks form. The conventional run is fitted with the – (ADA) augmentation, but we used a fixed augmentation and one with none. In any case and dataset, the lowest FID scores were achieved by those that used ADA augmentaiton. In fact, no augmentation starkly raised the FID scores.

For the third round of training runs, we considered different datasets to conduct transfer learning from. This is arguably the most important change that could

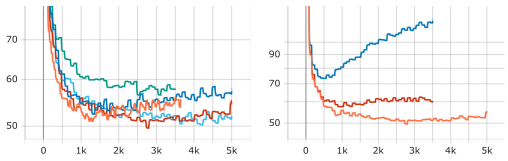

Figure 9. Caption

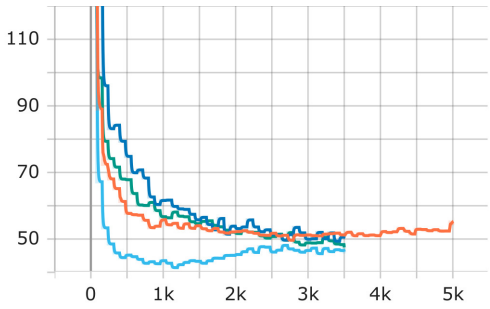

Figure 10. Caption

be made because —. Although images from the beach sunset dataset were relatively well defined and on-topic, we overlooked the variance of foreground elements dataset's images and the difficulties that this creates for training the GAN models. Considering that these images were scraped from social media, many users will both take photos with a subject posing in the foreground, and many will also take a photo of just the beach. Instead, we examined training with new domains that most consistently . Keeping the same parameters as the best run for beach sunsets, using a dataset of betta fish performed particularly well. At its minimum FID score, many generations of the images appeared near photo-realistic. Figure [].

Lastly, we performed transfer learning using a different starting networks. We used the other two pre-trained NVIDIA GAN networks at our disposal, Metfaces for 1024x1024 images, trained on portraits from the Metropolitan Museum of Art, and FFHQ for 1024x1024 images, trained on human faces from Flickr. For each network, we resumed training on a dataset of fighter jets and latte art. It was to be seen whether these datasets which the generator of the AFHQ network had trouble generating would perform differently from these new networks. Figure [] are the FID results. Across each network, the performance does not significantly vary. This interesting result indicates that regardless of what domain a network was previously trained on, the performances are similar for this new fighter jet dataset. In other words, there are particular datasets that can effectively be transfer-learned on many such networks, not just one.

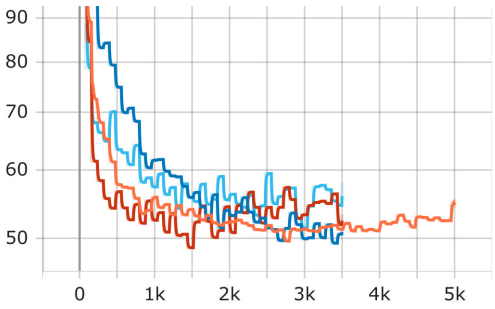

Figure 11. Caption

Figure 12. An image generated by generator

4. Conclusions and Future Work

In the 10-week period over which this project was conducted some conclusions about transfer learning on GANs can be made. Transfer learning stands a promising method to train generative adversarial networks using small datasets. Depending on the nature of the new dataset being used as well as the pre-trained network, high-quality images can be generated from datasets of under 2,000 examples. Additionally, viable datasets for training can indeed be sourced from Instagram. Implementing effective scraping and prepossessing tools can gather suitable images for training.

In training, many networks —.

Further research is certainly required to make more definitive conclusions about this topic. For one, specific properties and limitations of a limited dataset have yet to be well-defined. It is unknown how many examples exist for a dataset to be considered limited. The content of the image data can also play a role in how effectively a GAN learns, but there do not ex-

ist any metrics for that. Additionally, we have only used NVIDIA-made styleGAN3 networks. The transfer learning experiments conducted may give drastically different results when training on different GAN architectures that may exist. Lastly, without sufficient resources and a much larger dataset, it is still unclear the actual differences that exist between training networks from scratch and transfer learning.

References

[1] Vincent-Pierre Berges, Isabel Bush, and Pol Rosello. Text generation using generative adversarial networks. *sthalles.github.io*, 2017.

[2] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *sthalles.github.io*, 2018.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *sthalles.github.io*, 2014.

[4] 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. *sthalles.github.io*, 2017.

[5] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaako Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *sthalles.github.io*, 2021.

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *sthalles.github.io*, 2019.

[7] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. Controllable text-to-image generation. *sthalles.github.io*, 2019.

[8] Lars Ruthotto and Eldad Haber. An introduciton to deep generative modeling. *sthalles.github.io*, 2021.

[9] Thalles Santos Silva. A short introduction to generative adversarial networks. *sthalles.github.io*, 2017.