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 [7]. 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-

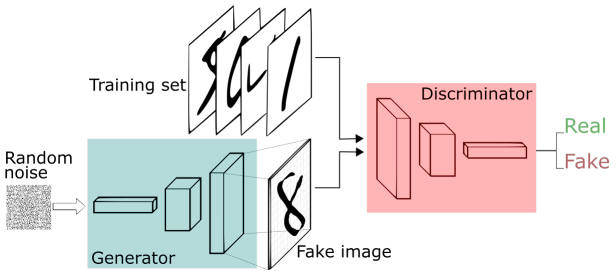

Figure 1. Basic generative adversarial network structure [9]

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 [4], and text-to-image generation [6].

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 [4]. 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 significantly, 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 such small datasets have been known to result in training failures [2]. Most commonly, insufficient data causes the discriminator to overfit, where the network improves in classifications faster than the generator can learn, giving unhelpful feedback to the generator and cause it to start generating random nonsense [8]. The chances of these occurrences can be minimized by using a pre-trained network and transfer learning.

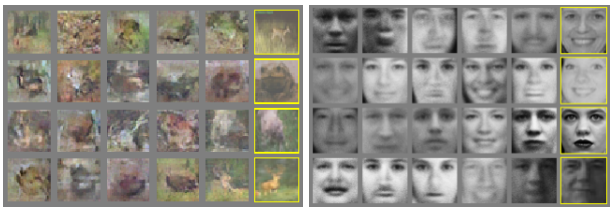

Figure 2. Left to right: Images generated by generative adversarial networks from Goodfellow et al. training on CIFAR-10 and TFD datasets [3]

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].

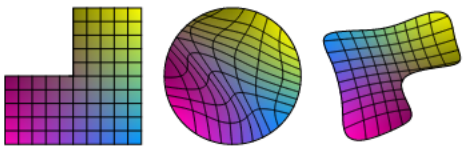

Figure 3. An illustraiton of the effects of the mapping network [5]. Note that the distribution in c) reflects that of the original in a) much more familiarly.

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 [5].

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) [5].

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 [5].

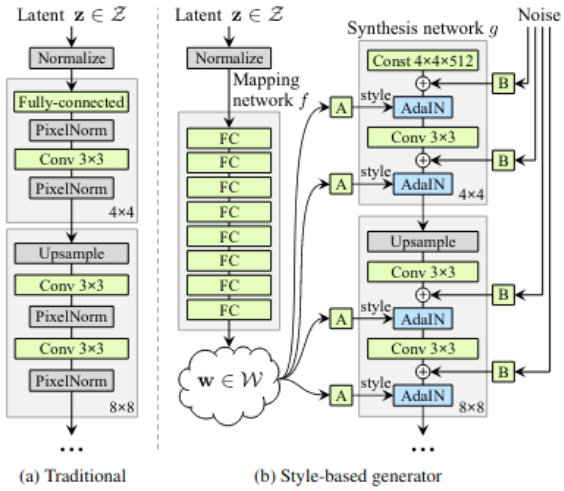

Figure 4. Traditional generator structure compared to that of styleGAN [5]. Note the use of a mapping network to create an intermediate latent space.

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, prominent text, human faces. The generator can have considerable trouble when attempting to generate the latter two elements of images in particular.

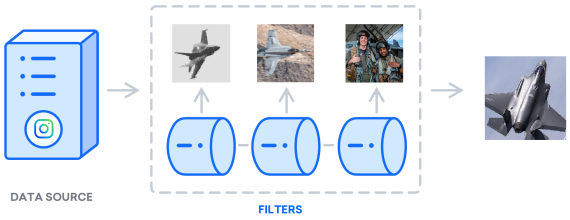

Figure 7. A Diagram of the current working data filtration pipeline. Various filters catch input images for criteria of unacceptable thresholds, leaving behind only usable images.

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. Manual discernment needed to be made 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 runs were conducted using the NVIDIA Corporation's official PyTorch ipmlementation of styleGAN3. Other Python implementations for dataset tooling and generation are also provided on the repository. —- computing cluster from Advanced Research Computing (ARC) at Virginia Tech. Three nodes of eight GPU's were alotted for this project at any given time. Using these computational resources, transfer learning training runs were conducted for between 3,500 and 5,000 steps. Training runs for 512×512 and 1024×1024 resolutions completed in around 15 and 19 hours, respectively. Performance was measured with the Fréchet Inception Distance (FID) score, which defines similarity between two image datasets. Lower FID scores correspond to greater resemblance between datasets, which would be used to compare real and generated datsets.

To find a training run to serve as a control, an initial round of training runs were conducted using NVIDIA's AFHQ GAN network pre-trained on a dataset of 16,130 512×512 images of animal faces. Training was continued on a limited dataset of 651 beach sunset images scraped from the Instagram hashtag "#beachsunset." Multiple runs were conducted to find the best hyperparameters to use on future runs. Adjustments were made to the generator and discriminator learning rates networks, R1 regularization on the discriminator, and the augmentation method. Combinations that resulted in stable training and the lowest FID scores would serve as the control run moving forward.

After setting hyperparameters, training runs using different datasets for transfer learning were conducted to examine how dataset domains influence training performance. Assessing differences in performance provides insight into specific attributes of image datasets that affect transfer learning and thus, its viability as a training method. On one hand, the exploration in the quality of synthesized images after training suggests the potential role that datasets play in the training process. In other words, certain datasets can be easy or difficult to train with, depending on the contents of the data. On the other hand,this might necessitate the need for manual intervention in the selection of datasets to be used for training. In this case, less is actually being learned, even if the training process performs strongly.

Lastly, training runs using different starting networks for transfer learning were conducted to explore the impacts of previous training on continued training on a new domain. Two other pre-trained NVIDIA GAN networks were readily available, the FFHQ and

Metfaces 1024×1024 networks. The FFHQ network was trained on 70,000 images of human faces from Flickr photos, and the Metfaces network was trained on 1,635 portraits from the Metropolitan Museum of Art. For various Instagram training datasets, training was continued on these two networks as was done using the AFHQ network. Evaluating training performance can suggest if datasets can be particular to a network.

4. Results

Images synthesized by the generator trained from the control run were decent and intelligible examples of beach sunsets. However, a sizeable portion of them were also disrupted by generations of chaotic shapes. It is thought that these additions result from high variability in the contents of the training dataset. Although most images from the beach sunset dataset were relatively well defined and on-topic, many of them still possessed prominent foreground features. Images from a social media website naturally feature these foreground features in the form of a person or a pet posing in front of the relevant hashtag material. Because this aspect of the dataset was overlooked, it appeared to upset the training process. leaving the generator confused on whether or not to generate foreground subjects without more training time or examples to improve learning.

Attempts to improve the performance were made through using different datasets to see if the domain had any impact. Images in new datasets were casually, but intentionally reviewed for subject relevancy and consistency in physical layout. Among these new datasets selected for transfer learning were betta fish, corgis, bald eagles, fighter jets, trains, mountains, and latte art. Figure [] features the results when transfer learning onto the small dataset of betta fish, an image synthesized by the generator network () and a generation snapshot using the model's wieghts (). The resulting synthesized images are much more realistic more consistently when training with this dataset compared to the dataset of beaches. In addition, although FID — . However, this does not mean that all new datasets performed as well. When conducting transfer learning on the dataset of latte art, the generator struggled to learn to synthesize elements of the image, as reflected by the nondecreasing FID score. In many generations,

Figure 9. Caption

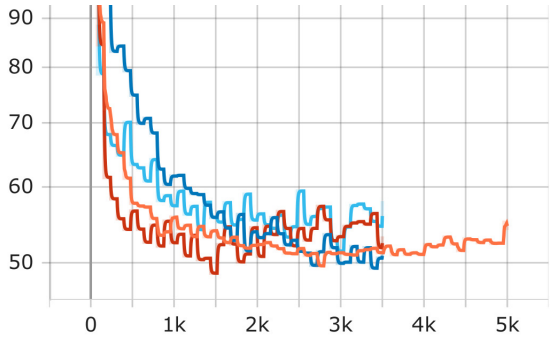

Figure 10. Caption

much of the image setting besides the coffee itself was chaotic noise, particularly the background. Based on these results, transfer learning can be a more or less effective method to train a GAN with depending on the dataset training is continued on.

Then, the Flickr Faces and Metfaces networks were used as the starting point for datasets that did not perform so well using the Animal Faces network to observe any change in performance when using a different pre-trained network. For both networks, training was resumed on a dataset of fighter jets, latte art, and bald eagles. It was found that performance did not significantly vary across any given dataset and that the resulting generations were of similar quality for each dataset. Figure [] shows a similar oscillatory state being approached for FID. This similarity demonstrates that the dataset is robust to the starting network. A dataset that performs well when training from one network is not particular to that dataset. Because this was the case for each dataset, it can be hypothesized that in transfer learning, the starting network does not bear much influence in the training process.

Additionally, it was found that a data augmentation technique is required for any transfer learning run. Data augmentation is the method of artificially

increasing the number of training examples through random edits to the training examples in the form of translations, rotations, cropping, and coloring. With the Instagram-scraped datasets having sizes of around 500-1,500, there are not that many images to train from, and thus the discriminator tends to overfit. In every dataset or network, using augmentation techniques improved training performance. In particular, the NVIDIA-made adaptive discriminator augmentation (ADA) was the best-performing method. On datasets of these sizes, data augmentation is a necessity.

5. Conclusions and Future Work

Transfer learning is a promising method to train generative adversarial networks on small datasets. Examining generations from the experiments, realistic images can be generated from a GAN trained for 3,500 steps on datasets of less than 1,000 images. The quality of these generations can vary greatly depending on the dataset, so careful selection of a training dataset is necessary. Most generally, a GAN learns the best when training on images containing a clear subject and consistent physical layout. Variability in these will not necessarily derail the training process, but will increase frequency of generating chaotic output. Additionally, it is possible that transfer learning from a particular dataset is robust across multiple starting networks. Strong learning of image features is sufficient to then apply a new dataset and adapt the network to it. In addition, Instagram can be an effective method to yield small datasets for transfer learning although ones with more examples are preferred.

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 exist any metrics for that. Additionally, while a dataset , the opposite is not necessarily true. What causes such incompatibilities should be further investigated, possibly with other networks, as only NVIDIA-made styleGAN3 networks had been used. 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. 2017.

[2] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 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. *Advances in Neural Information Processing Systems*, 2014.

[4] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaako Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 2021.

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *Advances in Neural Information Processing Systems*, 2019.

[6] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. Controllable text-to-image generation. *Computer Vision and Pattern Recognition*, 2019.

[7] Lars Ruthotto and Eldad Haber. An introduciton to deep generative modeling. *GAMM-Mitteilungen*, 2021.

[8] Isha Salian. Nvidia research achieves ai training breakthrough using limited datasets. *NVIDIA Blogs*, 2020.

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