GAN-Based Image Data Augmentation

Stanford CS229 Final Project: Computer Vision

David Liu
Department of Mathematics
Stanford University
dliud@stanford.edu

Nathan Hu
Department of Computer Science
Stanford University
zixia314@stanford.edu

Abstract

Generative adversarial networks (GANs) are powerful generative models that have lead to breakthroughs in image generation. In this project, we investigate the use of GANs in generating synthetic data from the MNIST dataset to either augment or replace the original data when training classifiers. We demonstrate that training classifiers on purely synthetic data achieves comparable results to those trained solely on pure data and show that for small sets of training data, augmenting the dataset by first training GANs on the data can lead to dramatic improvement in classifier performance. We also begin to explore using GAN-generated data to recursively train other GANs.

1 Introduction and Related Work

Decribed by Yann LeCun, Director of AI Research at Facebook AI, as "the most interesting idea in the last 10 years in Machine Learning", Generative Adversarial Networks (GANs) are powerful generative models which reformulate the task of learning a data distribution as an adversarial game. A fundamental bottleneck in machine learning is data availability, and a variety of techniques are used to augment datasets to create more training data. As powerful generative models, GANs are good candidates for data augmentation. In recent years, there has been some development in exploring the use of GANs in generating synthetic data for data augmentation given limited or imbalanced datasets [1]. Aside from augmenting real data, there are scenarios in which one may wish to directly substitute real data with synthetic data —for example, when people provide images in a medical context, having a GAN as the "middle man" would grant confidentiality to parties providing the original data.

Additionally, it has been shown that GANs can translate images from one context to another in the absence of paired examples [2]. This has been used to combine datasets of similar images in different formats. One variant of GANs, CycleGANs, have been used to combine datasets of contrast and non-contrast medical images [3].

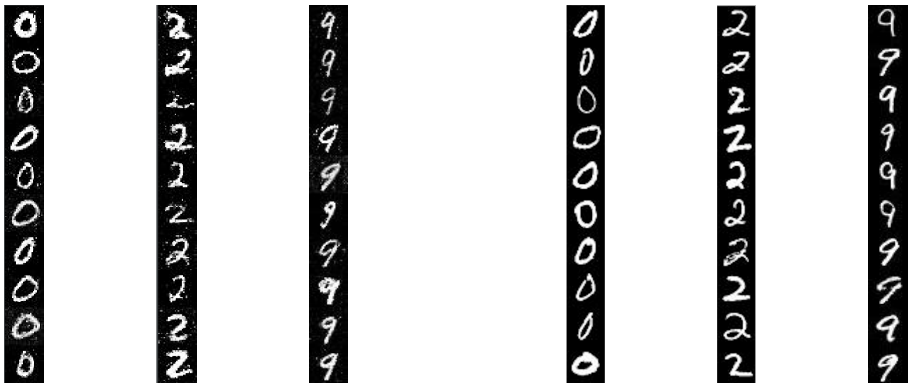


Figure 1: *Left Three:* GAN-generated numbers from the GAN with $\alpha = 4$ and training size of 2000 at 2000 epochs. *Right Three:* Real samples from the MNIST dataset.

The uses of GANs for data augmentation explored above generate low-dimensional numeric data or use CycleGANs to "translate" images, as opposed to generating truly new images from noise. However, GANs are known to be able to successfully generate outputs far more complex, such as realistic human faces [4]. This suggests that GANs have the potential to successfully augment more complex datasets, such as images.

Although the amount of data needed to train generative models may make their application to data augmentation impractical, other research has shown that a generative model can be learned from as little data as a single image. [5]. The "SinGAN" learns the coarse features of the images first and gradually moves to fine features, making it possible to generate images of comparable quality and content to the original. This shows the validity of training generative models given a small amount of data.

Another technique used to overcome a shortage of data can be seen in the realm of machine translation. When given a small dataset of parallel sentences between a source and target language, translation performance was improved by first training a model on the dataset, using the model to translate additional sentences in the source language, and repeatedly adding the model-translated sentences into the training set to train a new model [6]. We aim to adopt a similar process of iteratively training GANs using GAN-generated data to augment the data used to train the next generation of GANs. We call this process recursive training.

We thus turn to task of training GANs on the MNIST dataset to test the validity of GAN-generated data to augment and fully replace the original data when used to train a simple classifier. Compared to the previous work using GANs to augment medical datasets, our project aims to tackle a more complex dataset (MNIST images as opposed to numeric medical data) and a more complex task (10-way classification as opposed to binary classification). Lastly, we explore the validity of the recursive training of GANs described above to further improve performance given limited training data.

2 Methods

In this section, we discuss our GAN objectives and the model architectures that we use for our tasks. All of models we describe in the following subsections are built from scratch.

2.1 GANs

We trained a separate GAN to generate images of each digit. When training GANs, the generator and discriminator are playing a two-player minimax game [7] with value function

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

where $G : \mathbb{R}^{100} \to \mathbb{R}^{784}$ maps a noise sample to a 28×28 image and $D : \mathbb{R}^{784} \to \mathbb{R}$ maps an image to a probability that the image came from the true data distribution (rather than the generator). The discriminator tries to maximize the objective function, so its loss function over all examples is

$$J_D(\boldsymbol{x}, \boldsymbol{z}; \theta_d, \theta_g) = \frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

The generator can't affect the first term in the summation, so it tries to minimize the objective function by minimizing its loss

$$J_G(\boldsymbol{z}; \theta_d, \theta_g) = \frac{1}{m}\sum_{i=1}^{m}\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$

When training a GAN, we ascend the discriminator's stochastic gradient and descend the generator's stochastic gradient with respect to θ_d and θ_g, respectively, until the losses don't change anymore. The size of the hidden layers in each GAN are directly proportional to a parameter α. In order to combat overconfidence from the discriminator, we use one-sided label smoothing, which penalizes the discriminator for predictions for real images which exceeded .9. To implement this, we simply replace every instance of $\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$ with $\left(.9 - D\left(G\left(z^{(i)}\right)\right)\right)$ in the equations above.

2.2 Model Architecture

Although convolutional layers are common when building neural networks for computer vision tasks, we elected to instead used fully-connected layers which are more flexible and can express more complex relationships than convolutional layers. Fully-connected layers having many more weights are more expensive to train, but this cost is offset by the relatively small size of the images used. For all modules, we use a dropout rate of 0.1.

2.2.1 Generator

Images were generated starting from a 100-dimensional vector of noise drawn from a standard Gaussian distribution. We used 3 hidden layers with LeakyReLU (with negative slope .01) non-linearity and dropout, and output a 784-dimensional vector which encodes an image. Letting $\boldsymbol{h}^{[\mathbf{0}]}$ denote the input noise, $W^{[j]}$ and $\boldsymbol{b}^{[\boldsymbol{j}]}$ denoting the weight matrix and the bias vector in the j-th hidden layer, we have

$$\boldsymbol{h}^{[\boldsymbol{i}]} = \text{LeakyRELU}(W^{[i-1]}\boldsymbol{h}^{[\boldsymbol{i-1}]} + \boldsymbol{b}^{[\boldsymbol{i-1}]})$$

for $i = 1, 2, 3$. The size of each hidden layer were designed to grow exponentially between layers while parameterized by α, with $\boldsymbol{h}^{[\boldsymbol{i}]} \in \mathbb{R}^{16\alpha 2^i}$ and we output the vector $\boldsymbol{o} \in \mathbb{R}^{784}$ via

$$\boldsymbol{o} = \tanh(W^{[3]}\boldsymbol{h}^{[\mathbf{3}]} + \boldsymbol{b}^{[\mathbf{3}]}).$$

2.2.2 Discriminator

The discriminator takes in an image in the form 784-dimensional vector We again use 3 hidden layers with LeakyRELU and dropout, and output a probability that the image is legitimately from the dataset. Letting $\boldsymbol{h}^{[\mathbf{0}]}$ denote the input image, $W^{[j]}$ and $\boldsymbol{b}^{[\boldsymbol{j}]}$ denoting the weight matrix and the bias vector in the j-th hidden layer, we have

$$\boldsymbol{h}^{[\boldsymbol{i}]} = \text{LeakyRELU}(W^{[i-1]}\boldsymbol{h}^{[\boldsymbol{i-1}]} + \boldsymbol{b}^{[\boldsymbol{i-1}]})$$

for $i = 1, 2, 3$. The size of each hidden layer were designed to decay exponentially between layers while parameterized by α, with $\boldsymbol{h}^{[\boldsymbol{i}]} \in \mathbb{R}^{256\alpha 2^{-i}}$ for $i = 1, 2, 3$. and we output the prediction that the image comes from the original data $\boldsymbol{p} \in (0, 1)$ via

$$\boldsymbol{p} = \sigma(W^{[3]}\boldsymbol{h}^{[\mathbf{3}]} + \boldsymbol{b}^{[\mathbf{3}]}).$$

2.3 Classifier

The classifier uses a single hidden layer of size 300 and a sigmoid non-linearity to output a 10-dimensional vector representing how likely an image is to be a certain number. Letting p denote this prediction vector and the input image be $\boldsymbol{i} \in \mathbb{R}^{784}$, we have

$$p = W^{[1]}\sigma(W^{[0]}\boldsymbol{i} + b^{[0]}) + b^{[1]}$$

For prediction, the largest entry in the output vector corresponds to classifier's prediction. The loss function used is the cross entropy loss with an additional regularization term preventing the model from over-fitting. When trained on the entire MNIST training set, this classifier is able to achieve over 96 percent accuracy [8].

3 Experiments

3.1 Data

We used a random subset of the MNIST dataset for each of our experiments. Our largest experiments used 20000 of the 60000 total available training images due to GPU limitations. It's not crucial to achieve state-of-the-art results (of which the accuracy on MNIST is over 99.8% [9]), since we just want to show the efficacy of synthetic data in training a GAN.

3.2 Evaluation Method

We evaluate the quality of our GAN-generated synthetic data by training the simple classifier described above, using 10,000 GAN-generated images (1000 of each number). and evaluating that classifier's accuracy on the 10,000 images in the complete standard MNIST test set. Classifier training was done using an dev-set of 10,000 MNIST images.

3.3 Experimental Details and Results

For all of the experiments, the GANs were trained using Adam optimizer with different learning rates for the discriminator (.00075) and generator (.00015). Each training iteration of the GAN consisted of two steps of training for the generator and only 1 for the discriminator. GANs were trained for 2000 epochs. These asymmetries between the training process of the discriminator and generator were chosen to guarantee the generator and discriminator training losses equilibrate.

Due to the high cost of training GANs each GAN was only trained once. However, classifier evaluation was done by taking the average of 5 trials.

3.3.1 Experiment 1 - Classifier Training on Fully Synthetic Data

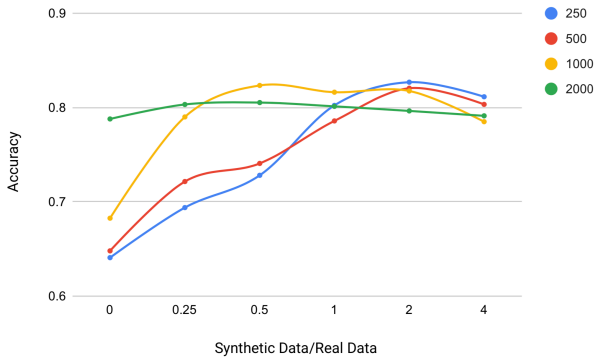
To begin, we explored how GANs of various sizes would perform when trained on differing amounts of data. The resultant GANs were used to generate synthetic data to train a classifier as described above and let us examine how a classifier would perform when trained on solely synthetic data

GAN Train Size	Baseline	$\alpha = 1$	$\alpha = 2$	$\boldsymbol{\alpha = 4}$	$\alpha = 6$
250	0.64076	0.42192	0.61512	**0.70920**	0.69808
500	0.64796	0.61074	0.74132	**0.76318**	0.70966
1000	0.68258	0.66974	0.69358	**0.75662**	0.73762
2000	0.78781	0.68692	0.68000	**0.79304**	0.78060

Table 1: Accuracies of classifiers trained solely on GAN-generated data

Train size represents the number of images of each number used to train each GAN. The baseline performance is that of a classifier trained solely on the original set of images. Despite intuition that smaller models may outperform larger models when given less training data, the accuracies in Table 1 show that on all sizes of training data, GANs of size $\alpha = 4$ outperformed all other models. Thus, we decided to further evaluate mixed datasets between real and synthetic data for the models with $\alpha = 4$.

3.3.2 Experiment 2 - Data Augmentation of Small Datasets



	Train Size			
Mixing Ratio	250	500	1000	2000
0	0.6408	0.6480	0.6826	0.7878
0.25	0.6938	0.7215	0.7810	0.8032
0.5	0.7281	0.7407	**0.8233**	**0.8052**
1	0.8022	0.7857	0.8162	0.8012
2	**0.8268**	**0.8204**	0.8175	0.7963
4	0.8113	0.8032	0.7849	0.7911

Figure 2: Analyzing mixing ratio vs. accuracy.

We proceeded to test the validity of using GANs to augment data. Using GANs of size $\alpha = 4$, the original images used to train the GANs were supplemented by GAN-generated images. This mixture of original and synthetic images were then used to train the classifier. As in the previous section, each classifier's accuracy shown in Figure 2 reflects the average performance across 5 trials. The mixing ratio = $\frac{\text{Synthetic Data}}{\text{Real Data}}$ reflects the amount of GAN-generated data used to augment the original data.

3.3.3 Experiment 3 - GAN Recursive Training

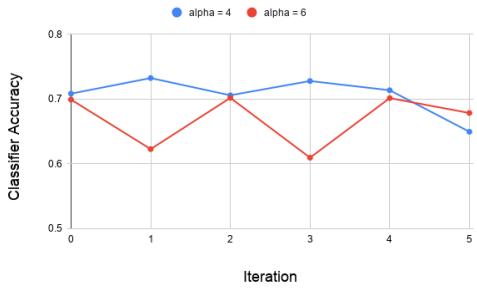


Figure 3: Recursive Training Accuracies

Lastly, because GAN-generated data is successful in augmenting the training data for training classifiers, we tried repeatedly using GANs to augment the dataset. This was done (for each number) by starting with a training set of $s_0 = 250$ real images. During each iteration i, the current training set of size s_i was used to train a GAN, which was in turn used to generate $0.1s_i$ synthetic images. These images were then added to the training set and used in subsequent iterations. The GAN size was kept constant across iterations, but in each iteration, the GANs were trained anew. To evaluate each generation of GANs, classifiers were trained on solely GAN-generated data as in Experiment 1. The classifier accuracies of each generation of GANs are shown in Figure 3.

4 Analysis

From Experiment 1, we observe that classifiers trained on purely synthetic data can achieve comparable and sometimes better results than classifiers trained directly on MNIST images. This suggests that in situations where privacy concerns make disclosing real data undesirable, GAN-generated data represents a comparable alternative to the original data. In addition, we note that regardless of train size, the GAN of size $\alpha = 4$ outperformed other model sizes. This suggests that when optimal GAN hyper-parameters are more influences by the variability and nature of the training data rather than the quantity.

Experiment 2 showed that adding synthetic data to real data increases classifier performance for all train sizes (see Table 1). For small training sizes of 250, 500, the classifier performance increases by over 15 percent when adding GAN-generated images. For all training sizes, we observe a point at which incorporating additional synthetic data decreases classifier performance. This optimal ratio between synthetic to real data differs based on the original train size, and decreases as the training size increases. In addition, classifier performance when given a train-size of 2000 was barely impacted by data augmentation. These two observations are in line with the general intuition that data augmentation is more beneficial when there is less data. Perhaps most surprising is the observation that incorporating GAN-generated data can improve classifier performance more than incorporating additional real images. For $s = 250, 500, 1000$, we observe that classifier performance when trained on s real and s synthetic images exceeds classifier performance when trained on $2s$ real images, despite the synthetic numbers being generated from GANs trained on the s real numbers and thus not containing any additional information about the dataset. In Figure 4, the classifier losses when training a classifier on 500 real and 500 synthetic images of each number vs. training on 1000 real images are shown. Both trials used the same number images for the same duration, but training on a mixture of original and synthetic data shows a far less variable and smooth descent of the training loss. This different behavior could be a result of the GAN-generated images having less variability than real images; then, the greater variability and noise in the training loss of the classifier on real data would reflect the greater variability in the images it is being trained on.

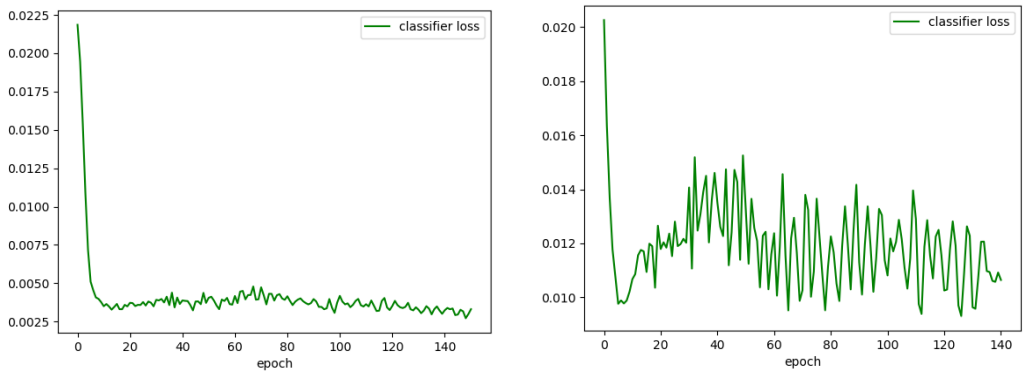


Figure 4: Classifier losses for Experiment 2. *Left:* 500 real and 500 synthetic numbers; *Right:* 1000 real numbers

For Experiment 3, we don't observe any significant increase in performance as we train for more iterations. However, the change between iterations wasn't monotonic either; rather, it appears periodic. Since the only real data the GAN is trained on are the original 250 images, it's not unreasonable for the accuracy to be stagnant because it never gets more information about the distribution of the data. The oscillating accuracy is likely due to the noisiness of the new synthetic data, and if we kept running the experiment for longer, we would expect the accuracies to drop, just as they did in Experiment 1 when too much synthetic data was added.

5 Conclusion

Our findings show training classifies solely on GAN-generated data can produce comparable performance to that of classifiers trained on the original data. This suggests that using GANs as an intermediary is a viable way to work with personal or private data. We also show that use GAN-generated data for augmentation can significantly improve classifier performance for augmenting small datasets. We even show scenarios in which the performance gain of adding GAN generated data exceeds that of adding more true images. However, additional tests suggest that we cannot benefit from recursively augmenting the training data of additional GANs, for classifiers trained on self-training GANs never seem to do better than after the single-iteration mix.

The primary limitation that our study faced was the limited computational power; due to limited access to GPU, we had to train everything on local machines. This ended up being quite costly and limited the rate at which we could train, and will only get worse if we intend to scale up to more complex images. If given access to more computational power, it would be worth doing similar classification on more complex datasets, such as CIFAR-10 or CIFAR-100. Additionally, some GANs clearly had better generator losses than other due to the differing complexity and variability of each number. Some future work could include training each GAN with a different set of hyperparameters in order to have the generators for different digits be more consistent. In addition, exploring data augmentation using other generative models - such as variation auto-encoders - both in isolation and in conjunction with GANs could be interesting. Lastly, there's more work to be done on self-training (perhaps in the form of a hyperparameter sweep, or increasing model size as the dataset increases), and with more time, we would have liked to investigate in more detail how a recursive approach to training compares to a one-shot approach.

6 Contributions

Both team members contributed equally to the project and report. The majority of the actual modules in the code (classifier, GANs, training scripts) and the design of experiments were a joint effort, and Nathan built the data processing pipelines while David built the plotting and model loading pipelines.

7 Code

The repository for this project is available at `https://github.com/dliud/gan`.

References

[1] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. Data Augmentation Using GANs. In *Proceedings of Machine Learning Research*, 2019.

[2] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *International Conference on Computer Vision*, 2017.

[3] Veit Sandfort, Ke Yan, Perry Pickhardt, and Ronald Summers. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. In *Nature Research*, 2019.

[4] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN. 12 2019.

[5] Tamar Rott Shaham, Tali Dekel, and Tomar Michaeli. SinGAN, Learning a Generative Model from a Single Natural Image. In *International Conference on Computer Vision*, 2019.

[6] Jiajun Shen, Peng-Jen Chen, Matt Le, Junxian He, Jiatao Gu, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. The source-target domain mismatch problem in machine translation. 09 2019.

[7] Ian Goodfellow et al. Generative Adversarial Nets. In *Conference on Neural Information Processing Systems*, 2014.

[8] CS 229. Problem Set 4. 2020.

[9] Adam Byerly, Tatiana Kalganova, and Ian Dear. A Branching and Merging Convolutional Network with Homogeneous Filter Capsules. In *arXiv*, 2019.