Semi-Supervised Learning Using GAN

Abstract

Modern machine learning is data intensive in nature. However, not all applications are appropriate enough to supply sufficient data. This study compares the performance of Generative Adversarial Network (GAN) against different dataset by setting the number of real input data as a control variable. In addition, based on the result the study trains the generator with feature matching. This forces the discriminator to learn features of the labeled set which match with the features of the generated samples, thereby reducing over-fitting to labeled set.

1 Introduction

Since the era Artificial Neural Network gains its popularity in machine learning, researches have been utilizing the superior speed of computers to process the overwhelming amount of data following by relative simple algorithm. Hoping the machine can be "trained" and produce intelligent results. However, some questions can not be solved by this approach naturally due to the limited amount of data can be acquired. For example, analysis of world-famous artworks, diagnosis of rare lethal cancers, restoration of unique historical artifacts. Hence, this study analyzed the current implementations of GAN [1] with the focus of reducing its data-intensive requirement [2]. Hopefully, the research can set a guideline of how well the GAN react to the limited amount of training data and propose a few different ways of improving its efficiency by given constraints.

1.1 Preliminary Works

As per [3] Training GANs consists of finding a Nash equilibrium to a two-player non-cooperative game. Each player wishes to minimize its own cost function, J(D) (Θ(D) , Θ(G)) for the discriminator and J(G) (Θ(D) , Θ(G)) for the generator. A Nash equilibrium is a point (Θ(D) , Θ(G)) such that J(D) is at a minimum with respect to Θ(D) and J(G) is at a minimum with respect to Θ(G). Unfortunately, finding Nash equilibrium is a very difficult problem. Algorithms exist for specialized cases, but we are not aware of any that are feasible to apply to the GAN game, where the cost functions are non-convex, the parameters are continuous, and the parameter space is extremely high-dimensional. The idea that a Nash equilibrium occurs when each player has minimal cost seems to intuitively motivate the idea of using traditional gradient-based minimization techniques to minimize each player's cost simultaneously. Unfortunately, a modification to Θ(D) that reduces J(D) can increase J(G) , and a modification to Θ(G) that reduces J(G) can increase J(D) . Gradient descent thus fails to converge for many games.

There have been several works adapting GANs for semi-supervised learning. One approach is to change the standard binary discriminator of a standard GAN to one that predicts class labels of labeled examples while enforcing the constraint that generated data should result in uncertain classifier predictions. There is also another approach which uses the discriminator of the GAN as the final classifier but instead modifies it to predict K + 1 probabilities (K real classes and the generated class). We have tried to implement feature matching as generator loss and also incorporate the complement generator idea from [2]. Instead of using a CNN network for discriminator and classifier, we tried to build the network using only fully connected layers.

2 Objective

- To classify accuracy loss of the GAN, discriminator particularly, when gradually reduce the number of real training examples
- To improve the accuracy of the discriminator given the limited amount of real training examples, from hundreds to a few tens

3 Materials and Methods

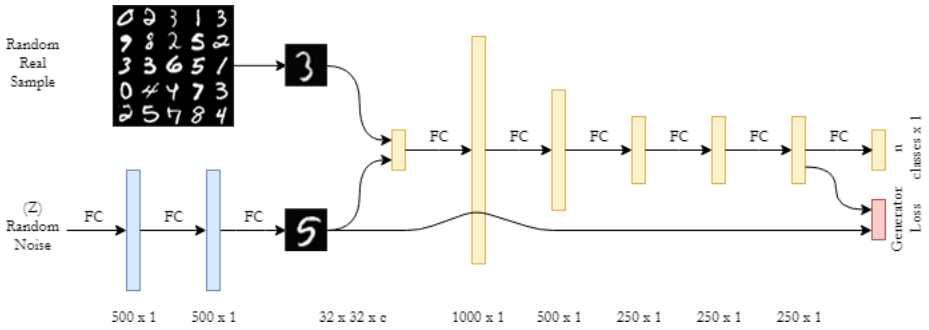

Figure 1: GAN based on feature matching technique. The generator error is directly computed by difference between the output of the generator and the output of the second last layer of the discriminator.

A vanilla GAN is constructed based on Mehta's work [1]. It consists of two major part: Generator who generates fake images based on seeded random noise and Discriminator who receive images and output 1.) if the image is fake; 2.) what class is the image belongs to. The vanilla GAN is used for accuracy benchmark and comparison against our proposed improvements. The structure of the network is consistent along the experiment while only the number of real samples are changing during the benchmark. Then the study updated this structure to the one figure 1 illustrated to be more robust for fewer real samples.

The study analyzed the MNIST, SVHN, and CIFAR10 datasets. All of them are downloaded from the PyTouch official repository. For reduced sample experiments, each dataset is shuffled based on a seeded random variable. After normalization, the top number of needed samples are acquired from the dataset. Later, they are passed to uniformed transformation and normalization process to augment the samples. All samples are in 32(w) x 32(h) x 3(c) for SVHN and CIFAR10, 32(w) x 32(h) x 1(c) for MNIST. Instead of the standard convolution pipeline, The discriminator network uses a series of fully connected layers with linear weight normalization, ReLU activation and adding seeded random noise in each layer of the network. The generator network is also a series of fully connected networks with softplus activation and batch normalization in each layer.

Generator network is trained on discriminator loss between the fake images produced by the generator and un-labeled images used for training. This technique is called as feature matching. The idea behind feature matching is rather simple - similar representations have similar statistics. Feature matching is an alternative objective that forces the generator to produce fake images that in the discriminator have representations similar to the ones of real images. The images generated by this generator are unrealistic fake samples around the high-density region. As per [2], this generator is called as complement generator which implies that the generator is a function of the perfect generator

(produces fake realistic images, where the generated image distribution perfectly matches the true data distribution).

The supervised loss is calculated as the cross-entropy loss between predicted and ground-truth labels. According to Ming et al. [4], the unsupervised loss is regularized using entropy loss between predicted outputs from un-labeled and generated images. This regularization reduces the entropy and encourages the classifier to assign a definitive label to the input.

3.1 Feature Matching

Feature matching addresses the instability of GANs by specifying a new objective for the generator that prevents it from overtraining on the current discriminator. Instead of directly maximizing the output of the discriminator, the new objective requires the generator to generate data that matches the statistics of the real data, where we use the discriminator only to specify the statistics that we think are worth watching. Specifically, we train the generator to match the expected value of the features on an intermediate layer of the discriminator. This is a natural choice of statistics for the generator to match since by training the discriminator we ask it to find those features that are most discriminative of real data versus data generated by the current model.

3.2 Batch Normalization

Batch Normalization (BN) is a technique to accelerate the training of deep neural networks and has been shown to be effective in various applications. To summarize, BN takes a batch of samples x1, x2, ..., xm and computes the following: yi = xi - μB σB $\cdot$ Υ + β, where μB and σB are the means and standard deviations of the input batch and Υ and β are the learned parameters. As a result, the output will always have a mean β and a standard deviation Υ, regardless of the input distribution. Most importantly, the gradients must be back-propagated through the computation of μB and σB.

3.3 Weight Normalization

For a linear layer y = WT x + b, where x $\in$ R n, y $\in$ R m, W $\in$ R n$\times$m, and b $\in$ R m, weight normalization performs a reparameterization W with V $\in$ R n$\times$m and g $\in$ R m: wi = gi ||vi ||2 $\cdot$ vi, where wi and vi are the i-th column of W and V, respectively. As with BN, the computation of ||vi ||2 is taken into account when computing the gradient with respect to V. Although presented as a reparameterization that modifies the curvature of the loss function, the main idea is to simply divide the weight vectors by their norms.

4 Results

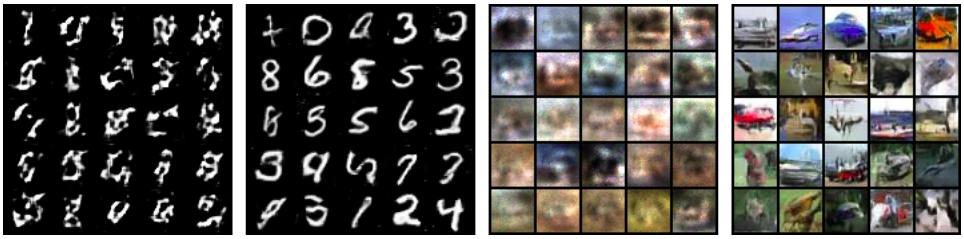

Figure 2: Results of original GAN and experimented GAN on MNIST and CIFAR10.

The study confirmed the GAN is generating meaningful results first. Above images show that the synthesized images from later iteration (right) are improved from earlier iteration (left) for MNIST and CIFAR10 accordingly. Note that the objects generated on the CIFAR10 do improve visually with more clearly defined contour separated from background rather than a blob of colors. However, the details are still lacking for a human to recognize what class the objects are.

Examples	CNN	SGAN	Ours
1000	0.965	0.964	0.985
100	0.895	0.928	0.980
50	0.859	0.883	0.985
25	0.750	0.802	0.919

Our works based on previous research [2] achieved higher classifier accuracy, especially for fewer samples. We also validated the results from Mehta's work [1] in the second column of the table. Note that feature matching in this case does not produce realistic results but does increase the performance of the discriminator. Fully connected performance for MNIST is good. But for CIFAR and SVHN when we decrease the number of labeled inputs, the accuracy decreases more significantly.

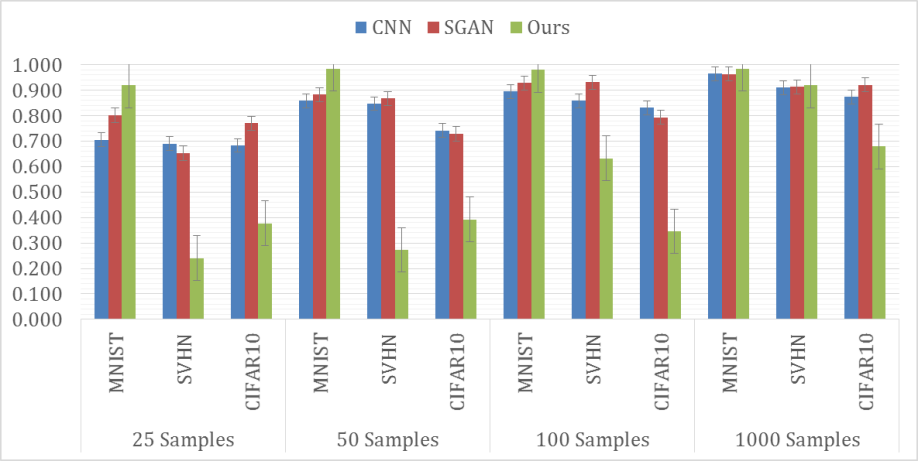

Figure 3: Comparison between different dataset and number of real samples.

The above figure shows the relationship of GAN discriminator accuracy vs a number of real samples passed to the network. Following the same configuration, the comparison is also made between different datasets and different augmentations. The trend shows the accuracy is going down while the number of real samples is reduced. However, with the help of various improvements the study introduced, the accuracy significantly increased when the number of real samples reduced to around 25. The study also discovered that more complex dataset, say CIFAR10, is less sensitive to the improvement. The reason is that with or without the improvement, discriminator of the dataset cannot reach a threshold to produce results good enough. I.e. the synthesized data is improving but still too few of them passed the discriminator to be considered "real" samples. Thus, the later augmentation has no data to work on along the pipeline.

5 Conclusions

The study successfully confirmed the efficiency results from Mehta's research while expanded the improvement from Dai et al. [2] to the more complicated dataset. The information acquired here forms a foundation for future study to answer a question how much performance drop is expected for a specific number of data available quantitatively. In addition, few guideline-temptations are made to improve efficiency when available data are insufficient. For future study, related problems for more complicated dataset should be focused. The study also suggests a more direct way to measure the performance of the generator can help to resolve the issues. Currently, the generator performance can be reflected by the discriminator accuracy. However, the discriminator is not a ground truth checker and it cannot be applied to more general cases to compare results from different discriminators.

References

[1] Mehta, R. (2018, April 03). Semi-Supervised Learning and GANs. Retrieved from https://towardsdatascience .com/semi-supervised-learning-and-gans-f23bbf4ac683

[2] Dai, Z., Yang, Z., Yang, F., Cohen, W. W., & Salakhutdinov, R. (2017, November 03). Good Semi-Supervised Learning that Requires a Bad GAN. Retrieved from https://arxiv.org/abs/1705.09783

[3] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016, June 10). Improved Techniques for Training GANs. Retrieved from https://arxiv.org/ abs/1606.03498

[4] Ding, M., Tang, J., & Zhang, J. (2018, October 22). Semi-Supervised Learning on Graphs with Generative Adversarial Nets. Retrieved from https://arxiv.org/pdf/1809.00130.pdf

Reference to our GitHub: https://github.com/MachinelearningSFU/SemiSupervisedGAN