Effects of Optimizer Choice on Mode Collapse in GANs

Logan Vegna [1]

Abstract

GANs (Generative Adversarial Networks) are extremely useful for a wide variety of tasks, however due in part to their complexity they have many common issues such as mode collapse. This paper explores various optimizers with respect to mode collapse on GANs including ADAM, RMSProp, SGD, and SGD with momentum. This paper compares the ability of the optimizers to reduce the occurence of mode collapse, both through empirical methods as well as anecdotal methods.

1. Introduction

1.1. Generative Adversarial Networks

Generative Adversarial Networks (Goodfellow et al., 2014) are a subset of machine learning models which create synthetic data usualy in the form of images. GANs have a wide variety of uses including text to image generation, artificial face aging and photo editing. Like any other unsupervised learning problem GANs must learn how to generate the desired images using example data. To achieve this, two neural networks are used in conjunction: A discriminator which learns to discern between generated data and real data, and a generator which actually generates data based on random inputs (or seeds). The two neural networks are trained in conjunction. First the generator generates a novel set of images, then the discriminator classifies both the set of generated images and a set of real images, finally the two models are optimized based on the losses. The loss for the generator is calculated from the discriminators outputs on only the fake images while the discriminators loss is calculated based on only the real inputs.

1.2. Mode Collapse

Due to the use of a discriminator and generator network a unique training error can arise which is not possible for typical machine learning models: Mode collapse. Mode collapse is typically signified by a significant difference in the distribution of the classes of the examples generated compared to the training data. In extreme cases every image generated will closely resemble the same exact class as can be seen in Figure 1. However, in most cases of mode collapse the generator will still produce some varying examples.

Mode collapse occurs when the generator finds an example which is extremely good at tricking the discriminator and begins only producing that one example. This is because the learning objective of the generator is to do exactly that: find the one example which can best trick the discriminator. Ideally the discriminator should learn to always reject a generator output if it is being constantly repeated. However, in some cases the discriminator will become stuck in a local minima unable to change its strategy. In these cases the discriminator will fail to learn to reject repeated examples, thus reinforcing the tendency of the generator to create the same example.

Researchers tolerance of mode collapse varies based on the use-case of the GAN. For instance, a GAN which is used to supplement training data for other machine learning tasks has very little mode collapse tolerance since the synthetic data will no longer represent the true distribution of the data.

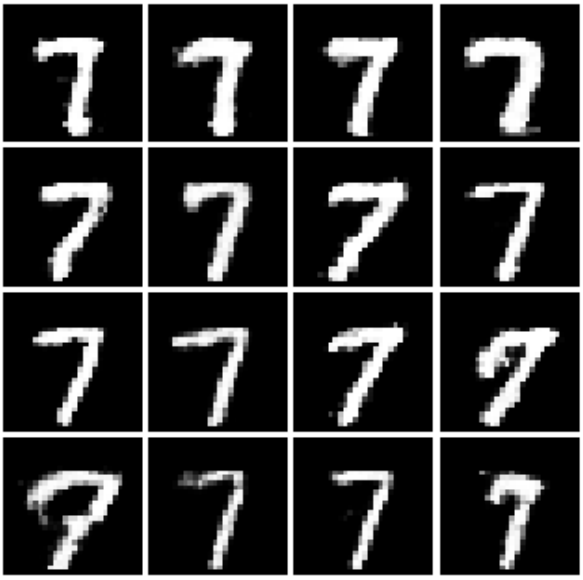

Figure 1. Example of mode collapse on the MNIST dataset. (Hui, 2017)

2. Optimizers

Optimizer choice plays a significant role in the degree which mode collapse occurs. This is due to the simple notion that mode collapse occurs because of inability of the optimizer to break out of local minima in non-convex optimization. Thus choosing an optimizer which has mechanisms to avoid getting stuck in local minima should reduce mode collapse.

The optimizers which will be explored in this paper are SGD, SGD with momentum (Rumelhart et al., 1986), ADAM(Kingma & Ba, 2014), and RMSProp (Tieleman & Hinton, 2012). Each of these optimizers has been chose due to there differing abilities to escape local minima. In theory SGD should be the worst at escaping local minima as it tends to bounce around minima while decreasing its learning rate. SGD with momentum should somewhat relieve this as the momentum paramater will help to accelerate the gradients towards the global minima. RMSProp works similarly to momentum in that accelerates the gradients toward the global minima, meaning it will be better at breaking out of local minima. However, it does so with the assumption that weights which have been recently updated a lot will not need to be updated as much in the near future. Finally Adam combines the best of both RMSProp and Momentum being able to adapt gradients in different directions and be as fast as momentum.

3. Methods

3.1. Dataset

The dataset chosen for the experiments is MNIST (Deng, 2012). MNIST is an optimal dataset to test for mode collapse on since it has a relatively small number of classes and does not require extreme amounts of training. The images were scaled to 28x28 pixels with one color channel. MNIST has sixty thousand examples with ten classes (0-9) which follow an approximately even distribution.

3.2. Model

All methods used in this paper were implemented using the TensorFlow Keras package

3.2.1. Generator

The generator model uses three two dimensional convolutional transpose layers which sequentially compound to upscale the input into a 28x28x1 output image. A LeakyRelu (Maas, 2013) activation function is used between each layer except for the output layer which utilizes tanh. The input used for the generator is a tensor of one hundred float32s.

3.2.2. Discriminator

The discriminator model is much simpler than the generator model and is based on a simple CNN (Krizhevsky et al., 2012). It has an input shape of 28x28x1 to match the output of the generator. The model uses two 2d convolutional layers with 64 and 128 filters of size 5x5 and a stride of 2. LeakyReLU is used for the activation function of all layers.

3.3. Training

For the training step cross entropy loss (Zhang & Sabuncu, 2018) was used to update the weights for both the generator and the discriminator. A somewhat small batch size of 256 was used such that there would be an opportunity for the model to experience mode collapse. Each model ran for fifty epochs. All the tested optimizers were initialized with a learning rate of .001 and a momentum of .5 for SGD with momentum.

3.4. Evaluation

To evaluate the generators performance Inception Score (Salimans et al., 2016) and Kullback-Leibler Divergence (Kullback & Leibler, 1951) were used. Kullback-Leibler Divergence is a measure of how close one distribution is to another distribution. A direct measure of mode collapse severity is achieved by comparing the distribution of labels produced by the generator and the true distribution of labels in MNIST. In this case a score of 0 indicates there is no mode collapse and a score higher than 0 is proportional to the amount of mode collapse. Inception score combines Kullback-Leibler with a measure of how realistic the examples the generator produces are. Inception score provides a good understanding of how well the model is performing overall or how well a human might consider the model to be performing. Opposite to KL, the higher the IS score the better the generated images are and the less mode collapse is occuring. In addition to the two empirical evaluation metrics the results of the GAN were measured qualitatively. This was simply performed by generating a set of 64 images and subjectively ranking the quality and mode collapse severity on a scale of 1 to 10 through observation.

To utilize the two aforementioned methods the approximate label distribution of the generator must be determined. To do so a small CNN was used which was trained to classify images in the MNIST dataset, ultimately reaching 99 percent validation accuracy. Then 500 examples were generated by the GAN and classified by the CNN to produce the approximate label distribution.

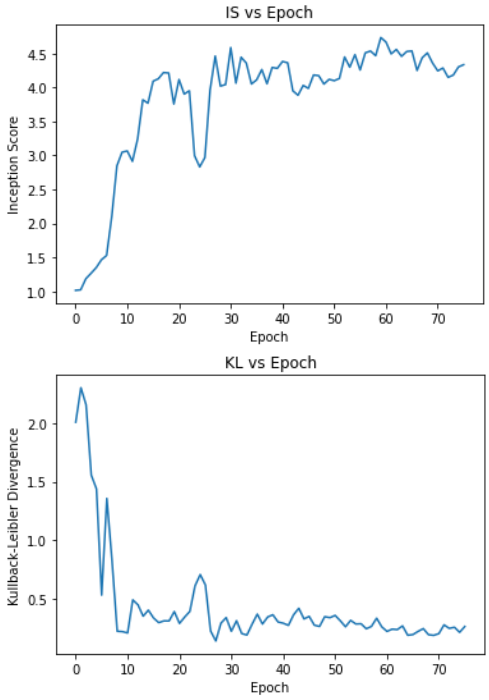

Figure 2. SGD Inception Score and Kullback-Leibler Divergence, and Mode

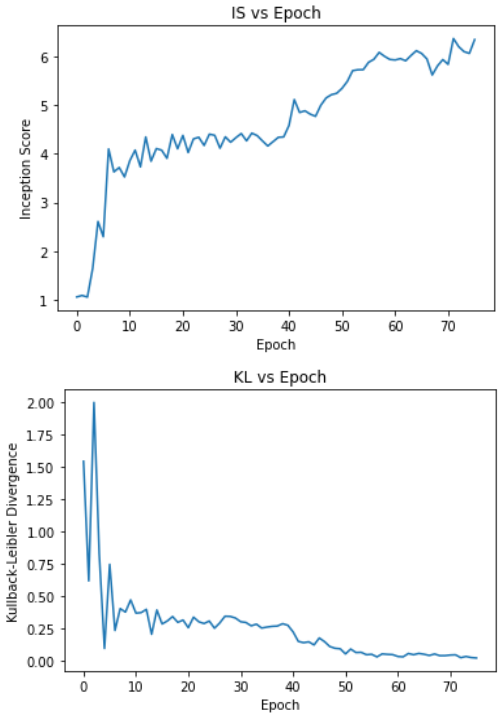

Figure 3. SGD with Momentum Inception Score and Kullback-Leibler Divergence

4. Results

4.1. SGD

Of the four models tested SGD had by far the worst Inception Score, Subjective Quality, Kullback-Leibler Divergence, and Mode Collapse Severity. Due to the lack of a robust mechanism to escape local minima the SGD optimized model found itself experiencing significant mode collapse, still producing only 8s, 9s, 3s and 0s after 75 epochs. This is no surprise since small variances in an 8 will result in number looking like 9s, 3s, and 0s whereas other numbers need larger variations to appear. From the charts in Figure 2 it can be observed that the generator quickly found an example which the discriminator was somewhat content with around epoch 10. Unfortunately, it was never effectively rebuffed by the discriminator for sticking with just that example and continued producing them until termination. The lack of image quality compared to the other optimizers can be explained by the relatively small learning steps normally characteristic to SGD optimizers.

4.2. SGD with Momentum

SGD with momentum performed significantly better than SGD however still had some minor issues with mode collapse and quality. Ultimately the images were distinct numbers however were somewhat sloppy, and there was a subtle absence of 4s and 6s. Interestingly, it can be observed from the charts in figure 3 that early on SGD with Momentum got trapped in a local minima similar to the SGD run. However, SGD with momentum was able to escape this local minima after only 40 epochs and begin converging on the global minima. This is a clear indicator of the momentum mechanism working to avoid permanent entrapment in local minima and avoiding mode collapse.

4.3. Adam

Ultimately Adam achieved nearly identical results as SGD with Momentum in terms of the evaluation metrics however, there are some difference with how those results were achieved. It can be clearly seen in figure 4 that Adam never experienced the same minima or interim mode collapse that SGD and SGD with Momentum did. This is indicated by the lack of any plateau in the evaluation metrics. Instead, the model's IS and KL continuously improved, closely resembling a logarithmic function. This indicates that it would be extremely unlikely for ADAM to experience severe mode collapse. It is likely that the quality of the images produced by both Adam and SGD with Momentum would continue to improve by a small amount with more epochs.

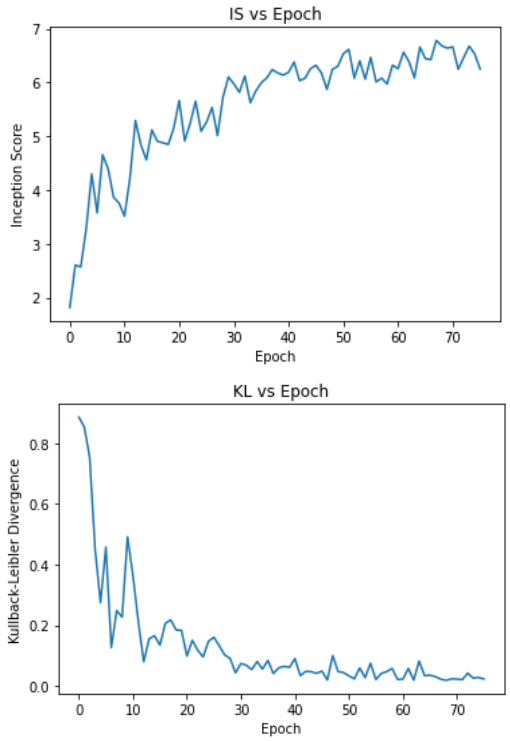

Figure 4. Adam Inception Score and Kullback-Leibler Divergence

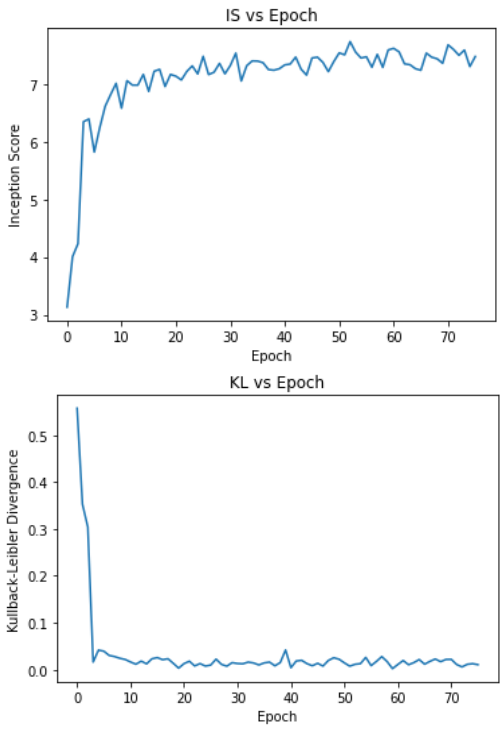

Figure 5. RMSprop Inception Score and Kullback-Leibler Divergence

Table 1. Evaluation metrics at termination for each optimizer.

OPTIM	IS	KL	QUALITY	MODE COLLAPSE
SGD	4.33	0.261	3/10	7/10
SGD MOMENTUM	6.35	0.018	7/10	3/10
RMSPROP	7.73	0.007	9/10	1/10
ADAM	6.25	0.023	7/10	3/10

4.4. RMSprop

Surprisingly, RMSprop outperformed both Adam and SGD with Momentum in all evaluation metrics, especially when one considers that Adam combines RMSprop and SGD with momentum. It can be seen from Figure 5 that RMSprop achieved extremely good results within just 30 epochs, after which minuscule gains were made as the optimizer bounced around the global minima for 45 epochs. Additionally the Kullback-Leibler Divergence plummeted to near zero in just 5 epochs. A possible explanation for the rapid convergence when compared to Adam is the lack of momentum. The use of momentum in Adam certainly should help it avoid local minima however its use results in the much slower decay of step size upon reaching a global minima. This results in the optimizer jumping around near the global minima but not precisely finding it for some time. Conversely once RMSprop locates a global minima it quickly decays and the step sizes become small resulting in higher precision.

5. Conclusion

From the results of the experiments it is clear that SGD should be completely avoided when training GANs. Using an SGD without Momentum will in the best case result in extremely slow convergence to the global minima, and in the worst case unacceptable levels of mode collapse and poor image quality. The major pitfall when using SGD for training GANs is its inability to accelerate towards the global optima or breakout of local minima, thus any optimizer which does not have these features should be avoided for GAN training. SGD with Momentum should also be avoided in most cases due to its observed behavior of plateauing at a local minima during training. If the GAN is particularly sensitive to experiencing mode collapse the addition of momentum onto SGD still may not be able to break out of a local minima.

On the other hand the results of the experiments indicate that RMSprop should be utilized over Adam. However, Adam is still a perfectly valid optimizer to use when training SGDs and may even work better than RMSprop on different models and datasets. However, due to its much quicker convergence to the global minima, RMSprop may achieve the same results or better as Adam in significantly less training time. Neither Adam nor RMSprop seemed at all

prone to mode collapse at any stage. Thus, when deciding on the optimizer for a GAN, if compute resources allow for it, it would be prudent to try using Adam if after a small number of epochs RMSprop has not rapidly resulted in high quality low mode collapse images.

References

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

Hui, J. Generative adversarial nets (gan) , dcgan, cgan, infogan, 2017. URL https://jhui.github.io/2017/03/05/Generative-adversarial-models/.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Kullback, S. and Leibler, R. A. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86, 1951.

Maas, A. L. Rectifier nonlinearities improve neural network acoustic models. 2013.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans, 2016.

Tieleman, T. and Hinton, G. T. tieleman and g. hinton. lecture 6.5, 2012.

Zhang, Z. and Sabuncu, M. R. Generalized cross entropy loss for training deep neural networks with noisy labels. In *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.