Proceedings of Machine Learning Research vol 75:1–3, 2018

On the Training of Generative Adversarial Nets

Mingrui Liu MINGRUI-LIU@UIOWA.EDU
Department of Computer Science, The University of Iowa, Iowa City, IA 52242

Abstract

This report presents results for standard GAN training on MNIST dataset, where different optimization algorithms are used, including SGD, ADAM, AdaGrad and another recently proposed algorithm called AmsGrad which is to address the issue in ADAM. The experimental results show that ADAM and AmsGrad perform considerably better than SGD, AdaGrad and SGD with momentum.

1. Experimental Setting

1.1. Network Structure

Both discriminator and generator use a neural network. Specifically, the architecture of discriminator consists of the following layers stacked together:

Fully Connected Layer (784×256)**, Leaky ReLU Layer with** $\alpha = 0.2$**, Fully Connected Layer** (256×256)**, Leaky ReLU Layer with** $\alpha = 0.2$**, Fully Connected Layer** (784×1)**, Sigmoid Activation Function.**

The architecture of generator consists of the following layers stacked together:

Fully Connected Layer (64×256)**, Leaky ReLU Layer with** $\alpha = 0.2$**, Fully Connected Layer** (256×256)**, Leaky ReLU Layer with** $\alpha = 0.2$**, Fully Connected Layer** (256×784)**, Tanh Activation Function.**

The Leaky ReLU is one attempt to fix the "dying ReLU" problem. Specifically, the function computes $f(x) = \mathbb{I}(x < 0)(\alpha x) + \mathbb{I}(x \geq 0)(x)$, where α is a small constant.

1.2. Setting During the Training Phase

In the original GAN paper (Goodfellow et al., 2014), the associated optimization problem during training is

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

In our implementation for training phase, we train the discriminator to minimize $-\log(D(\mathbf{x}))$, and train the generator to maximize $\log(D(G(\mathbf{z})))$ instead of minimizing $\log(1 - D(G(\mathbf{z})))$, as suggested in the original GAN paper. The batch size is set to be 100, and there are 200 epochs in total. We choose four popular algorithms, SGD (Nemirovski et al., 2009), SGD with momentum, AdaGrad (Duchi et al., 2011), ADAM (Kingma and Ba, 2014), and AmsGrad (Reddi et al., 2018). Please note that AmsGrad is recently proposed, in which the authors prove that the convergence analysis of ADAM in the convex case is wrong and provide a remedy (AmsGrad) to properly fix the issue in ADAM.

The parameter chosen for these algorithms are:

- For SGD without momentum, the learning rate is 0.0003;
- For SGD with momentum, the learning rate is 0.0003, the momentum is 0.9;
- For Adagrad, the learning rate is 0.0003;
- For ADAM, the learning rate is 0.0003;
- For AmsGrad, the learning rate is 0.0003.

Please note that the choice of learning rate is vital. We choose the best learning rate for SGD (with momentum) by grid search and uses the same learning rate for AdaGrad, ADAM, AmsGrad.

2. Experimental Results

2.1. Comparison between Real Image and Fake Image

For each optimization method, we compare the real image and the fake image generated at the last iterate. There are 5 Figures in total. It is easy to see that the fake images generated by Figure 4 and 5 are significantly better than that in Figure 1, 2, 3. It may imply that in practical perspective, ADAM and AmsGrad could be better than SGD, SGD with momentum and Adagrad when training standard GANs.

2.2. Other Criterion

We record the value of $D(\mathbf{x})$ and $D(G(\mathbf{z}))$. According to the theory, in the equilibrium state, $D(\mathbf{x}) = D(G(\mathbf{z})) = \frac{1}{2}$. We report these values for 5 algorithms at the last iterate at Table 2.2.

From the table, we know that ADAM and AmsGrad has better performance than SGD, SGD with momentum and Adagrad when training standard GANs, which is consistent with the phenomenon displayed in Figures.

Optimization Algorithms	$D(\mathbf{x})$	$D(G(\mathbf{z}))$
SGD	0.95	0.04
SGD+momentum	0.87	0.09
AdaGrad	0.83	0.10
ADAM	0.68	0.20
AmsGrad	0.84	0.28

Table 1: Comparison for Different Optimization Algorithms

3. Future Work

We plan to investigate the 5 Algorithms respectively for the Wasserstein GAN training (Arjovsky et al., 2017), and hope to give practitioners a guide to choose optimization methods when conducting deep learning experiments related to GAN.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

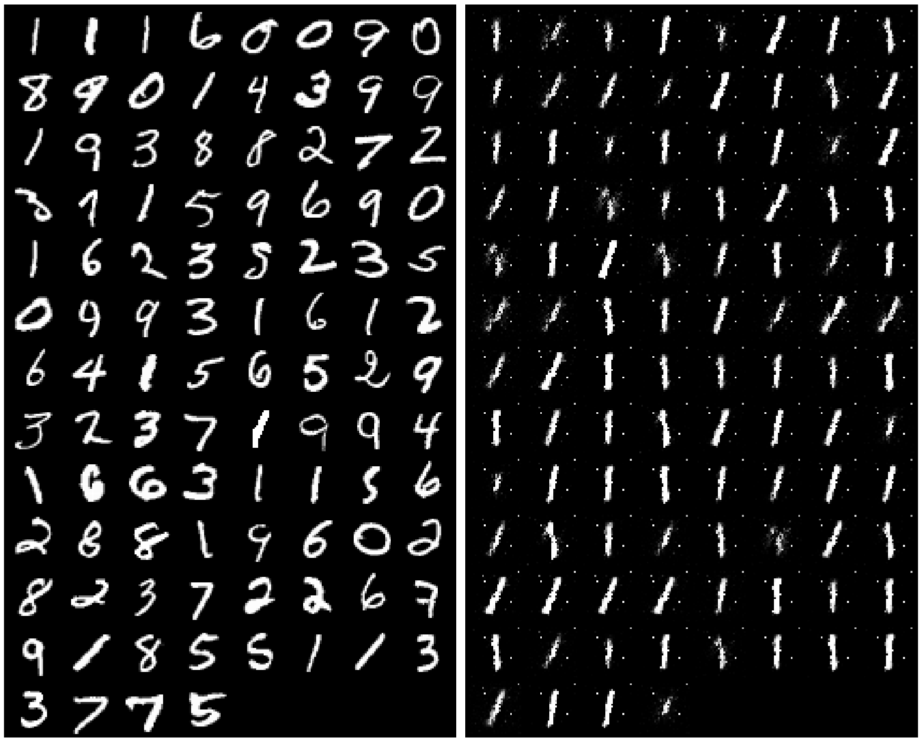
Figure 1: Optimizer: SGD, left: Real Image, Right: Fake Image

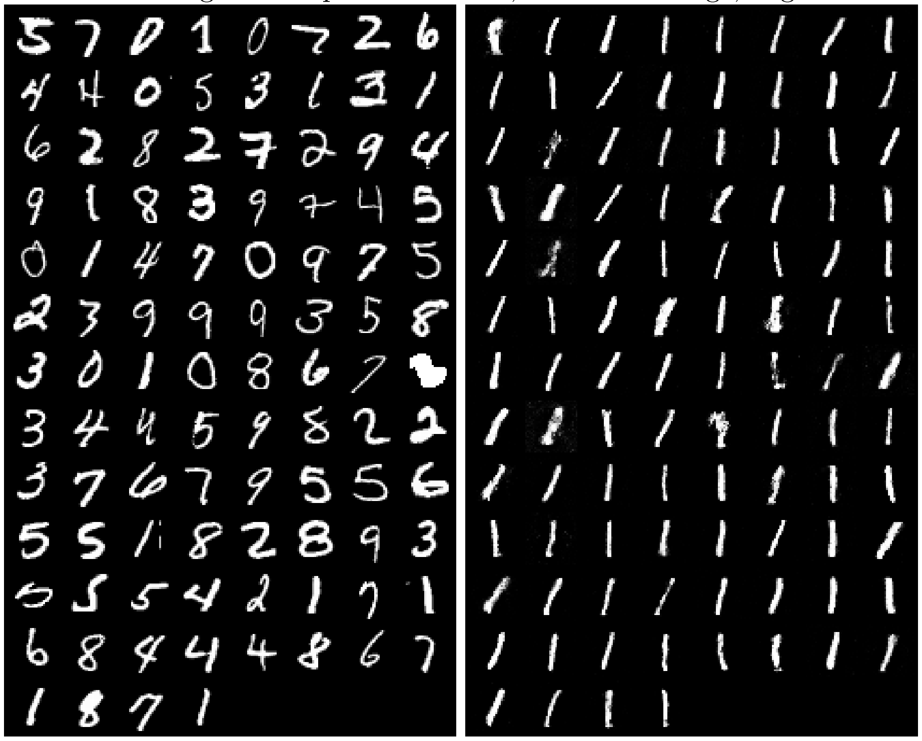
Figure 2: Optimizer: SGD with momentum, left: Real Image, Right: Fake Image

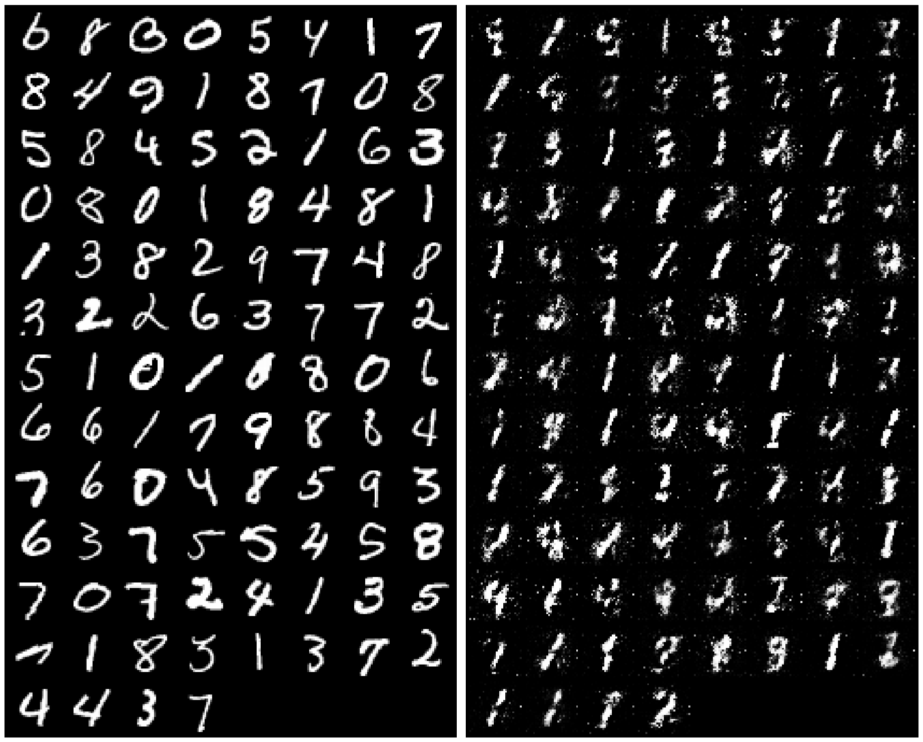

Figure 3: Optimizer: Adagrad, left: Real Image, Right: Fake Image

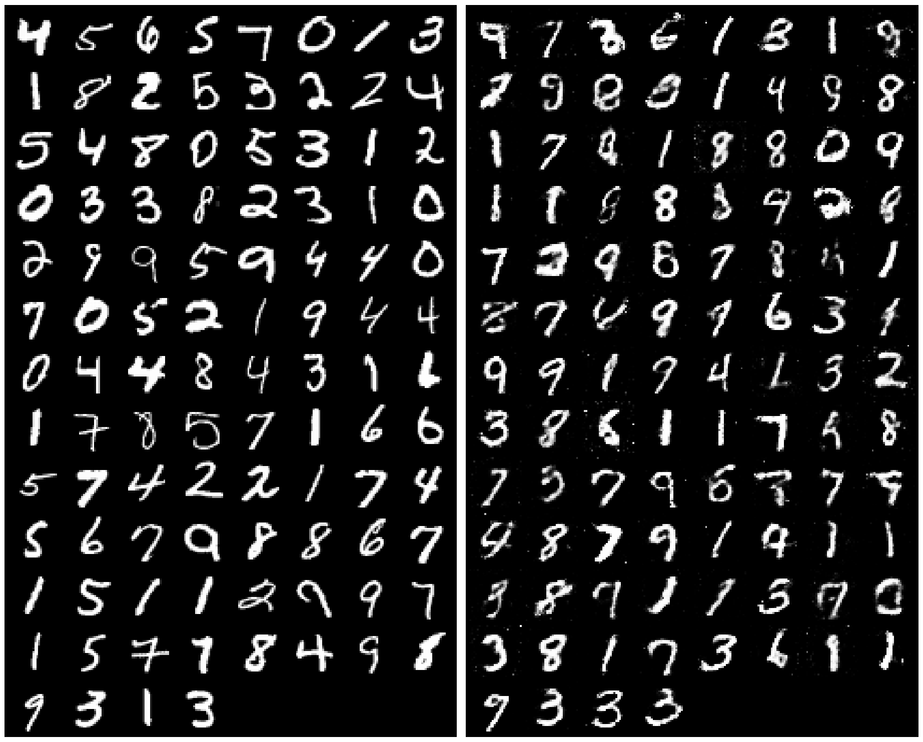

Figure 4: Optimizer: ADAM, left: Real Image, Right: Fake Image

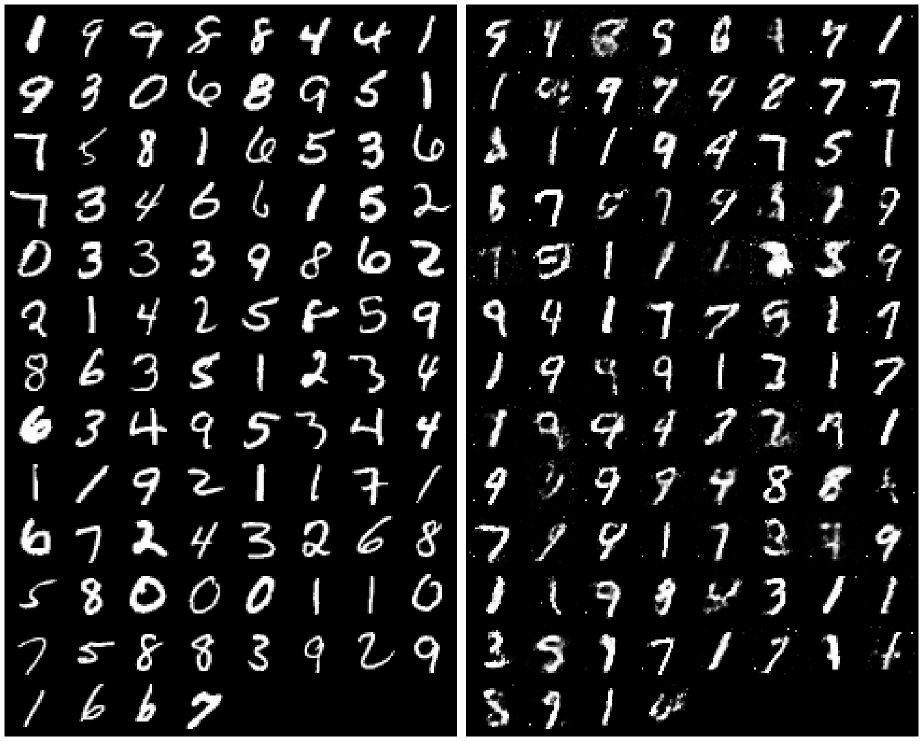
Figure 5: Optimizer: AmsGrad, left: Real Image, Right: Fake Image