Analysis of Gradient Descent Optimization Algorithms on ResNet

Chirayu Garg, Abhinav Garg, Anshu Raina
Department of Computer Sciences, University of Wisconsin-Madison
{cgarg2,garg26,araina2}@wisc.edu

Abstract

Gradient descent optimization algorithms like stochastic gradient descent (SGD) are most widely used to train neural networks. Recently, adaptive variants of gradient methods like Adagrad, Adam etc. have caught popularity and are being used by almost all the new architectures for training. Since these algorithms perform local optimization for each parameter, they are supposed to give better results. Some studies have shown that adaptive methods can give undue influence to spurious features, hence generalizing worse than the SGD. In this work, we analyze different variants of gradient descent algorithms to find out if the adaptive methods are really useful. Our results show that with some tuning of hyper-parameters, adaptive methods attain same accuracy as SGD on validation and test sets.

1 Introduction

Deep learning algorithms have achieved remarkable feats in fields like image classification and speech recognition. At the very heart of deep learning is the gradient descent which is one of most common way of optimizing neural networks. Past several years have seen a surge in the introduction of many versions of gradient descent algorithms which provide the promise of good convergence with faster training speed[6, 4, 7]. Majority of the these algorithms introduced are adaptive methods and Adam[6] has become the de-facto standard for these algorithms. But the generalization and out of sample behavior of these methods are poorly understood and they give no guarantee of generalization.

Recent work done by [9] shows that adaptive optimization techniques generalize much worse than stochastic gradient descent (SGD) even when the solutions have better training performance. The authors make a case for reconsideration of adaptive methods to train neural networks. In our project we wanted to see if such a claim holds true for the larger networks. We implement one of the most popular network used for image classification called ResNet and perform rigorous experiments with different algorithms. We observe that by properly tuning the hyper-parameters and implementing a step-sized decay of the learning rate in ResNet-20, adaptive methods like Adam and Adagrad can give similar results when compared to non-adaptive methods like stochastic gradient descent (SGD) and stochastic gradient descent with Nesterov momentum (SGDN) in terms of validation and test set accuracy.

2 Background

Gradient descent remains one of the most popular algorithms to optimize deep neural networks. Over the period of time, many algorithms have been introduced to optimize gradient descent. In this section we will discuss about these algorithms based on [8]. We will limit the discussion to the algorithms we use in our experiments.

2.1 Stochastic gradient descent

Stochastic Gradient Descent (SGD) is a variant of gradient descent in which the parameter update happens for every training example. Vanilla gradient descent computes the gradient of cost function for entire training dataset before performing one update and SGD gets rid of this redundancy by performing one update at a time.

$$w_{k+1} = w_k - \eta \bigtriangledown_w f(w_k : x^i; y^i) \tag{1}$$

2.2 Mini-batch gradient descent

Mini-batch gradient descent is a combination of vanilla gradient descent and stochastic gradient descent. Here, the parameter update is performed on a mini-batch of say n training example where n is the batch size. In this paper whenever we refer to SGD, we actually refer to the mini-batch gradient descent as it is very useful to use mini-batches to get the maximum utilization of the hardware resources.

$$w_{k+1} = w_k - \eta \bigtriangledown_w f(w_k : x^{i:i+n}; y^{i:i+n}) \tag{2}$$

2.3 Nesterov accelerated gradient

Nesterov accelerated gradient adds a special momentum term while calculating the gradient [7]. The gradient calculation is done by not looking at the current parameters but w.r.t. to the future position of the parameters:

$$v_k = \gamma v_{k-1} + \eta \bigtriangledown_w f(w_k - \gamma v_{k-1}) \tag{3}$$

$$w_{k+1} = w_k - v_k \tag{4}$$

2.4 Adagrad

Adagrad is an adaptive gradient-based optimization algorithm that adapts the learning rate to perform larger updates for infrequent parameters and smaller updates for frequent parameters [4]. For all the gradient descent methods mentioned above, all parameters are updated in a single update as every parameter has the same learning rate. Adagrad uses different learning rate for each parameter.

$$w_{k+1,i} = w_{k,i} - (\eta / \sqrt{G_{k,ii} + \epsilon}). \bigtriangledown_{w_k} f(w_{k,i}) \tag{5}$$

G is a diagonal matrix in which each diagonal element (i, i) is the sum of squares of gradients up to timestep k. ϵ is a smoothing term used for avoiding division by zero.

2.5 Adam

Adam computes the adaptive learning rates for each parameter as well [6]. It keeps the exponentially decaying average of past squared gradients v_k and it also stores the exponentially decaying average of past gradients m_k.

$$g_k = \bigtriangledown_{w_k} f(w_{k,i}) \tag{6}$$

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k \tag{7}$$

$$v_k = \beta_2 v_{k-1} + (1 - \beta_2) g_k^2 \tag{8}$$

m_k and v_k are the estimates of first moment and second moment of gradient respectively. It was found by the authors of Adam that m_k and v_k are biased towards zero mostly for the initial time steps and when the decay rates are small. So, the following bias corrected first and second moment estimates were introduced.

$$\hat{m_k} = m_k / (1 - \beta_1^k) \tag{9}$$

$$\hat{v_k} = v_k / (1 - \beta_2^k) \tag{10}$$

Below is the Adam update rule:

$$\hat{w_{k+1}} = w_k - \eta / (\sqrt{v_t} + \epsilon) . \hat{m_k} \tag{11}$$

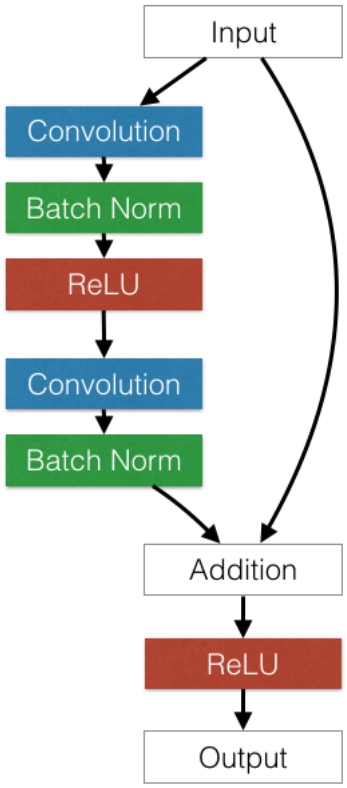

Figure 1: Basic Block of ResNet

3 Network architecture

The experiments for evaluating different gradient descent approaches are performed on Residual networks (ResNets)[5]. ResNets differ from the normal convolutional neural networks as they have a shortcut (skip) connection to each basic block which gets added to its output. Skip connections enable a clear path for gradients to propagate to early layers of the network which makes learning faster by avoiding the problem of vanishing gradients.

ResNet is a collection of multiple basic blocks which are serially connected together. In addition, there is shortcut connection to each basic block which gets added to output. Figure 1 shows the basic block of ResNet. The number of layers which we use in ResNet for our experiments is 20 as more number of layers take a lot of resources and time to train which was not feasible in the scope of this project. The datasets used for image classification are Cifar-10 and Cifar-100[1]. Input to the network is a 32x32 size image and the network learns a model to assign a class to a particular image.

4 Evaluation

As discussed, we perform our experiments for the aforementioned optimizers on ResNet-20. We implement our own ResNet in Tensorflow[3] with a step decay policy for learning rate in which the learning rate is reduced by a particular decay factor after a given number of epochs. With the number of resources we could get in a given amount of time, we try to be as much rigorous as possible in the selection of learning rate, step decay factor and number of epochs the algorithm runs before performing the decay. All these experiments are done on cloud lab machines [2]. These machines don't come with GPU support, so it takes us a lot of time to train a network as big as ResNet even if we restrict the number of layers to 20. The experiments were performed on both Cifar-10 and Cifar-100. Table 1 contains the parameters for best test accuracy of different algorithms on Cifar-10.

Table 2 has the best test accuracy parameters on Cifar-100. These parameters were found by doing multiple runs of the algorithms using different parameters.

Table 1: Best training parameters for different algorithms on Cifar-10

Algorithm	Learning Rate	Step decay	Step size
SGD	0.25	0.1	5
SGDN	0.2	0.5	10
Adagrad	0.1	0.5	5
Adam	0.02	0.1	5

Table 2: Best training parameters for different algorithms on Cifar-100

Algorithm	Learning Rate	Step decay	Step size
SGD	0.25	0.1	10
SGDN	0.15	0.1	10
Adagrad	0.25	0.2	10
Adam	0.01	0.1	5

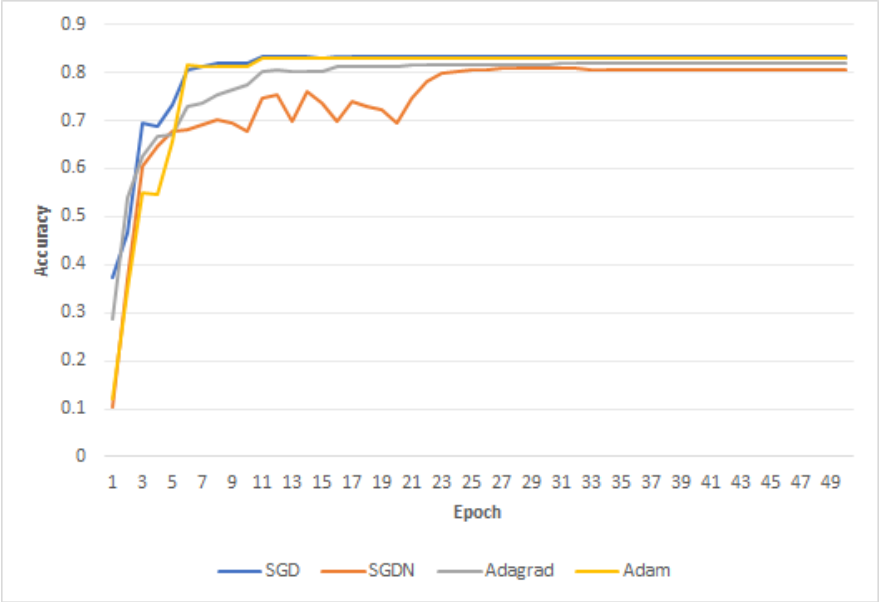

Figure 2: Top-1 Validation Accuracy Comparison for Cifar-10

Figure 2 and Figure 3 show the best validation and test accuracy of different algorithms on Cifar-10. As can be seen from the curves, all the optimizers reach the same accuracy after a particular number of epochs. Adam seems to converge faster for both validation and test set.

Figure 4 and Figure 5 show the best validation and test accuracy of different algorithms on Cifar-100. Again, all the optimizers reach almost same accuracy after a particular number of epochs. SGD and SGDN perform a bit better than Adam and Adagrad in terms of accuracy. Note that Cifar-10 and Cifar-100 both have 60000 images, but former has just 10 classes while the latter has 100 classes. So, the number of images per class is less for Cifar-100. It seems that if we have less number of examples per label, adaptive methods like Adam and Adagrad may converge to a lesser accuracy than SGD and SGDN though the difference is less.

In the experiments, we performed tuning to achieve maximum training and validation accuracy for both adaptive and non-adaptive methods. We found that non-adaptive methods require more tuning than adaptive methods. For best results, non-adaptive methods tend to have larger initial higher learning rate than the adaptive methods. Adam gives the best solution at a very small initial learning rate as compared to other methods.

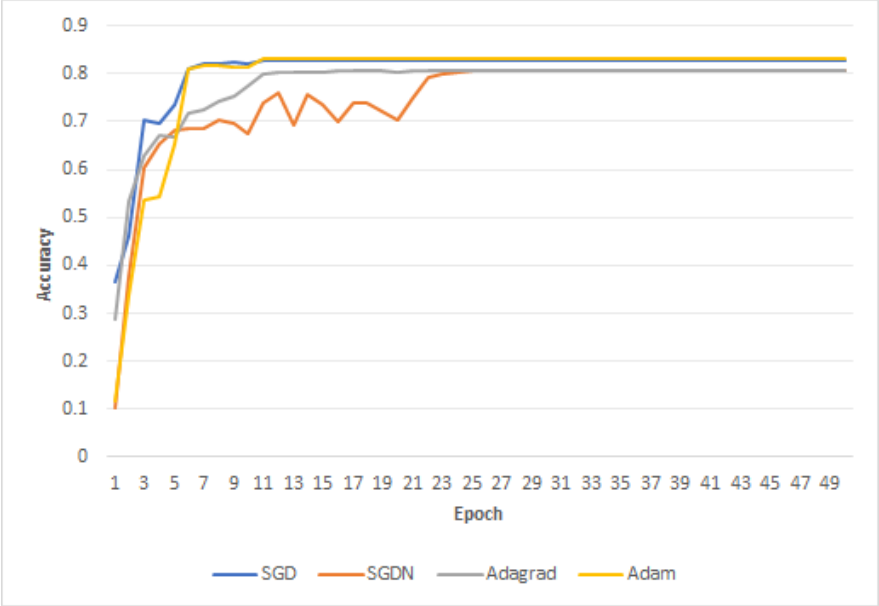

Figure 3: Top-1 Test Accuracy Comparison for Cifar-10

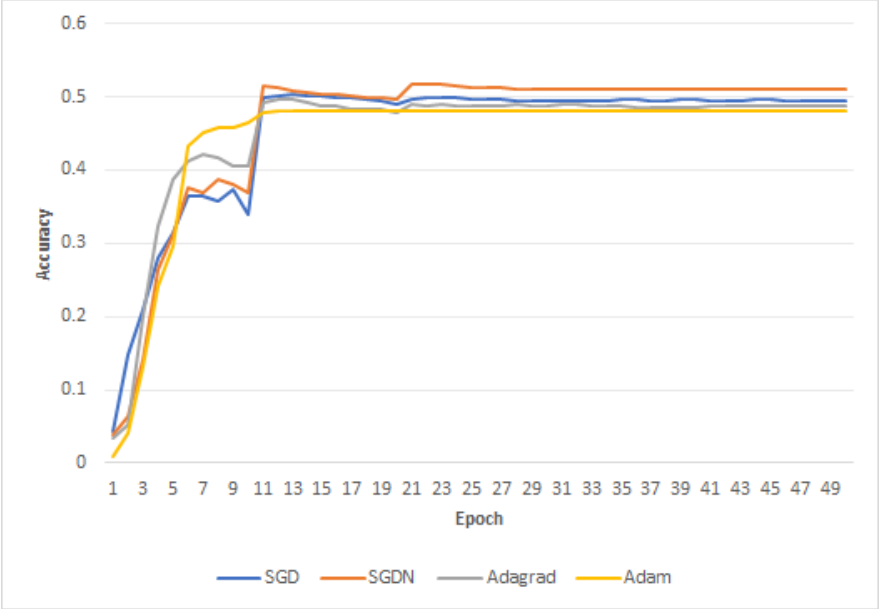

Figure 4: Top-1 Validation Accuracy Comparison for Cifar-100

5 Conclusion

The main motive of this project was to determine whether non-adaptive methods generalize better than adaptive methods. We find out that with proper tuning of hyper-parameters and implementing a step-sized decay of learning rate, adaptive gradient algorithms like Adagrad and Adam can achieve same accuracy as that of stochastic gradient descent algorithm with and without Nesterov momentum. Adam tends to reach convergence early for the given datasets.

Non-adaptive methods need more tuning to get best results than adaptive methods. So, if one doesn't want to do an extensive parameter search and just want neural network to learn faster, adaptive

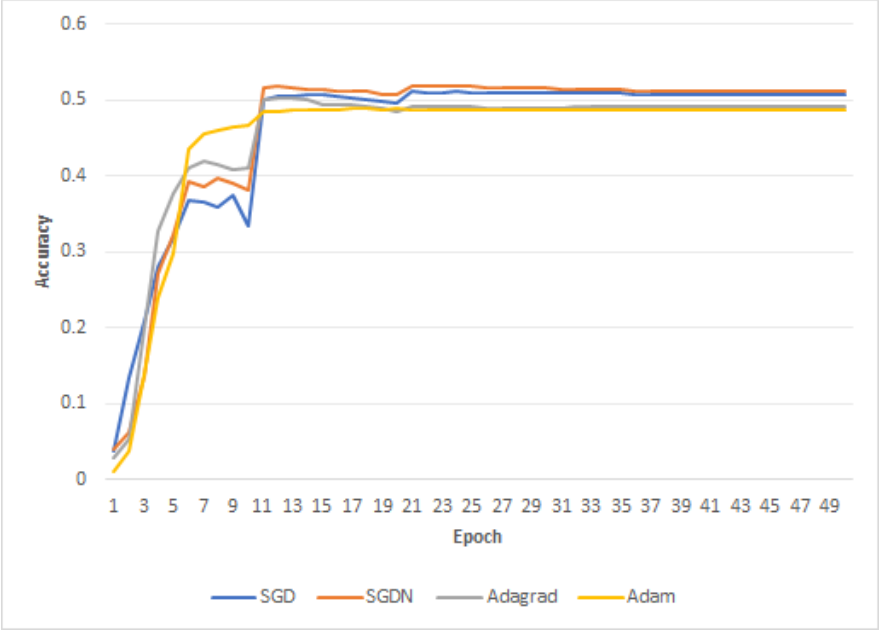

Figure 5: Top-1 Test Accuracy Comparison for Cifar-100

methods like Adam work better. But if one can afford to do an extensive hyper-parameter search, then a simple SGD or SGDN implemented with a weight decay may give similar or better results.

References

[1] Cifar-10 and cifar -100 dataset. `http://www.cs.toronto.edu/~kriz/cifar.html`.

[2] Cloudlab. `https://www.cloudlab.us`.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] 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.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[7] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547, 1983.

[8] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[9] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.