Adam vs. SGD: Closing the generalization gap on image classification

Aman Gupta AMAGUPTA@LINKEDIN.COM
Rohan Ramanath RRAMANATH@LINKEDIN.COM
Jun Shi JSHI@LINKEDIN.COM
S. Sathiya Keerthi KESELVARAJ@LINKEDIN.COM
Linkedin, Sunnyvale, CA

Abstract

Adam is an adaptive deep neural network training optimizer that has been widely used across a variety of applications. However, on image classification problems, its generalization performance is significantly worse than stochastic gradient descent (SGD). By tuning several inner hyperparameters of Adam, it is possible to lift the performance of Adam and close this gap; but this makes the use of Adam computationally expensive. In this paper, we use a new training approach based on layer-wise weight normalization (LAWN) to solidly improve Adam's performance and close the gap with SGD. LAWN also helps reduce the impact of batch size on Adam's performance. With speed in tact and performance vastly improved, the Adam-LAWN combination becomes an attractive optimizer for use in image classification.

1. Introduction

Adaptive optimization algorithms, such as Adam [11], have shown better optimization performance than stochastic gradient descent[1] (SGD) in some scenarios. However, recent studies (see below for details) show that Adam often leads to worse generalization performance than SGD for training deep neural networks on image classification tasks. The intent of the paper is to devise efficient ways of improving Adam to close the performance gap with SGD.

Wilson et al. [22] is one of the early efforts to show that adaptive methods do not generalize as well as SGD when tested on a diverse set of deep learning tasks. The authors also show that adaptive and non-adaptive optimization methods indeed find very different solutions with very different generalization properties on a special class of problems. Past research about Adam [2, 16, 22] can be summarized along the following:

A.1 Adam finds solutions that generalize worse than those found by SGD [3, 4, 6]. Even when Adam achieves the same or lower training loss than SGD, the test performance is worse.

A.2 Adam often displays faster initial progress on the training loss, but its performance quickly plateaus on the test error.

A.3 Learning rate and weight decay are important hyperparameters that must be tuned to do well on each dataset. In addition, one may consider tuning the three innate hyperparameters of Adam, β_1, β_2 and ϵ.[2] It has been found that the additional and expensive tuning of these three hyper-parameters significantly improves Adam.

Further research has been done to analyze and close the *adaptive generalization gap*[3]. Keskar and Socher [9] exploit A.2 to use Adam in the earlier stages of training but switch to SGD before the

1. All references to stochastic gradient descent include the use of momentum and weight decay.
2. β_1 and β_2 control the exponential decay rates of the moving averages of gradients (m) and squared gradients (v), and ϵ is used to deal with situations having very small v values.
3. Adaptive generalization gap is the difference in the generalization performance of adaptive optimizers and SGD.

learning saturates. They show that switching too late in the training process leads to a generalization gap similar to Adam. Choi et al. [2] work with A.3 to show that hyperparameters (that include β_1, β_2, ϵ) could be the reason that adaptive optimization algorithms fail to generalize. They use sophisticated hyperparameter tuning algorithms over a relatively large search space (see Appendix D of [2]) and conclude that the optimal parameters vary a lot between datasets. Choi et al. [2] make a strong claim that adaptive optimizers would never underperform momentum or SGD with sufficient hyperparameter tuning since they are a more general class of optimizers. Nado et al. [16] claim similar results after comprehensive tuning of Adam, and also study the effect of batch size. While these are important results to close the gap in our understanding of adaptive optimizers, they do little to improve the practical usability of Adam since it is prohibitively expensive to run the recommended number of training runs required to find the ideal hyperparameters for each dataset.

Models for image classification are trained with exponential-type loss functions like logistic loss and cross entropy that asymptotically attain their least value of zero when the network score goes to infinity. After the network has learned to correctly classify a large fraction of training examples, the weights and scores grow to make the training loss (and hence its gradient) very small. Let us refer to this as *loss flattening*; we will discuss its detrimental effects in §2. This is seen in optimizers like SGD [1] and Adam [11] when used with no (or mild) ℓ_2 regularization or weight decay. AdamW [14] improves on Adam by applying decoupled weight decay to ensure that the network weights don't get too large and loss flattening is delayed. The authors of [14] report that AdamW achieves comparable performance to SGD for small datasets like CIFAR-10 [12], but neither do they conduct experiments on large datasets nor do they study the effect of batch size on generalization performance. In addition to A.2, Adam's generalization performance decreases sharply as batch size increases because loss flattening occurs earlier.

Recently we proposed Logit Attenuating Weight Normalization (LAWN) [5], a training approach that begins with Adam in the initial warm-up phase and then fixes the weight norms of each layer for the remaining phase of training. Our claim in this paper is: LAWN, by avoiding loss flattening and increasing weight adaptivity, improves Adam to (i) make it on par with SGD, i.e., closes the adaptive generalization gap to zero; and (ii) makes Adam's performance degradation with batch size much milder.

In §2 we describe the details of LAWN and show empirically (§3) that adaptive optimizers work as well as SGD on image classification datasets even at large batch sizes.

2. Logit Attenuating Weight Normalization (LAWN)

Most deep neural networks used in practice are over-parameterized. When training with cross entropy loss and without weight decay, weights become large and the training loss goes to zero fast. The loss flattens towards zero and hence the magnitudes of gradients and hessians also become very small. In §1 we referred to this happening as loss flattening. Since Adam optimizes aggressively, loss flattening starts occurring quite early with Adam.

The main bad effect of loss flattening is that, with learning rates being limited in size, small gradients lead to small weight changes.[4] When this combines with small hessians, the ability to escape from a solution with inferior test error (around which training is currently residing) is severely hampered [21, 23]. We call this phenomenon as *the loss of weight adaptivity*. SGD, being

4. Though Adam uses $m/\sqrt{v}$ for updates, β_1, the decay factor for the first moment variable m is 0.9 while β_2, the decay factor for the second moment variable v is 0.999, making m go to zero much more rapidly than v.

much slower than Adam to reach points of loss flattening, ends up going to solutions with superior generalization performance.

The above is a key reason why weight decay [14] is used. It helps control the size of weights and thus avoids loss flattening. Thus, although regularization is typically understood from an overfitting perspective, we highlight its role in making the network more adaptive and enabling it to escape more easily from weights that generalize poorly. In a typical deep net design with weight decay, the three training hyper-parameters - weight decay parameter, peak learning rate, and learning rate schedule - are tuned to get the best performance. Though weight decay helps to improve adaptive optimizers, it is still not a good enough mechanism to bring adaptive optimizers on par with SGD.

An alternative way of controlling loss flattening is to constrain the magnitude of the weights. In [5] we proposed a specific training approach called LAWN, which we will briefly describe next. Since most layers of convolutional nets are homogeneous (ReLU activation [17]), it is appropriate to constrain the norms of the weight vector of each layer [5, 15]. LAWN begins the training of any given optimizer with standard initialization and without any constraints or weight decay. This is done for E_{free} epochs (E_{free} is a hyper-parameter) and it is called as the *free phase* of LAWN. After this free phase, each layer is forced to constrain its weight norm at its current value, and this constrained weight norm training is continued for the remaining epochs. This is the *constrained phase* of LAWN. While the free phase allows unconstrained movement of the weights to form gross classifier boundaries, the constrained phase, by avoiding loss flattening, allows the weights to be adaptive and finally settle in (flat) regions having superior generalization. The free phase can also be viewed as a natural way of choosing weight norm values for the various layers in the constrained phase.

From a design point of view, like standard training with weight decay, LAWN also involves just three hyper-parameters - the weight decay hyper-parameter of a typical design is replaced with the E_{free} hyperparameter. Also, E_{free} is a weak hyper-parameter; simply setting it to just a few epochs works well. So, overall solution wise, the cost of the full deep net design using LAWN is about the same as the standard training of optimizers using weight decay. Full details of LAWN can be found in [5]. In general, LAWN is useful in any situation where loss flattening causes performance loss.

As we will see in §3, when LAWN is used with adaptive optimizers, the lift in performance is substantial, allowing them to close the gap with SGD (with or without LAWN).

It is well known that increasing batch size worsens the performance of all optimizers [10]. This is attributed to the reduced ability to escape from a given (inferior) solution, which is caused by reduced stochasticity associated with increasing the batch size [23]. It is usual for both Adam and SGD to degrade performance quite badly when the batch size is increased. By avoiding loss flattening and thus keeping the escape energy sufficiently good, LAWN helps optimizers to perform better at large batch sizes. Empirical evidence suggests that the improvement helps Adam more than SGD at increased batch sizes.

The idea of constraining weight norms has also been considered in the works of Hoffer et al [8] and Heo et al [7]. But these methods do not have the free phase of LAWN. In Hoffer et al's method, the layer-wise weight norms are set at initialization using heuristics and then kept constrained at those values throughout training. The *AdamP* method of Heo et al [7] is slightly different and it allows the weight norm constraints to decrease a bit in each step based on the angle between the gradients and the weight vectors of the layers. LAWN is more powerful in improving Adam because of its free phase that sets the right values for the layer-wise weight norms. In Table 1 we compare AdamP and Adam-LAWN on CIFAR-10 and CIFAR-100 datasets and across two different values

of batch size. While both methods give similar performance on CIFAR-10, on CIFAR-100 AdamP significantly drops in performance at the large value of batch size while Adam-LAWN does not.

Method	**CIFAR-10**		**CIFAR-100**	
	256	10k	256	10k
AdamP	93.75	93.76	72.87	71.93
Adam-LAWN	93.91	93.84	72.99	72.97

Table 1: Test Acc. on CIFAR-10 and CIFAR-100. Standard error is in the the range of $[0.1, 0.45]$. On CIFAR-100, performance of AdamP drops significantly as batch size increases, but Adam-LAWN does not.

3. Experiments

We compare SGD [19], Adam and their LAWN variants on image classification tasks. Traditionally, SGD has out-performed Adam in this category. To demonstrate the efficacy of LAWN, we conducted experiments on the CIFAR [12] and ImageNet [3] datasets. All model training code was implemented using the PyTorch library [18] and experiments were conducted on machines with NVIDIA V100 GPUs. For each experiment, we report the average test metric over 3 runs. We did not tune the Adam hyperparameters ϵ, β_1 and β_2, which could have led to further improvements for Adam-LAWN.

The LAWN methods use a 3-phase learning rate schedule, while the original methods use a 2-phase learning rate schedule that incorporates both warmup and decay. For SGD and Adam optimizers, we tuned E_{warmup} (number of epochs for learning rate warm-up), peak learning rate, and weight decay. For the LAWN variants, we tuned E_{free}, E_{warmup} and peak learning rate. Details about hyperparameter tuning and learning rate schedule can be found in Appendix D of [5].

3.1. Image Classification for CIFAR-10 and CIFAR-100

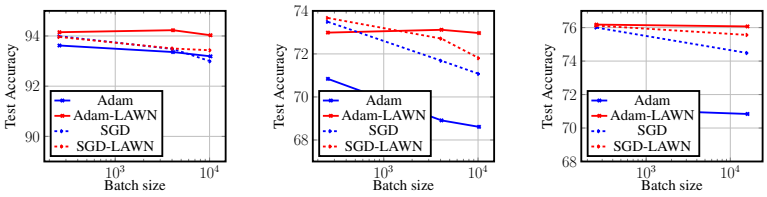
For both CIFAR-10 and CIFAR-100 [12], we used the VGG-19 CNN network [20] with 1 fully connected final layer. Our ImageNet experiments use a ResNet-based [6] architecture. All experiments were run with a 300 epoch budget. As seen in Table 2, LAWN variants either match or outperform the base variants across batch sizes. Adam-LAWN is particularly impressive. This is in stark contrast to earlier held beliefs that adaptive optimizers cannot match SGD's generalization performance for image classification tasks [22].

Effect of batch size. LAWN variants cause more graceful degradation of performance with batch size, as compared to base variants. Adam-LAWN causes almost no degradation in generalization performance even at batch size 10k (see Figures 1(a), 1(b)).

Effect of $\mathbf{E_{free}}$. We observed that switching early to LAWN mode (i.e. fixing E_{free} to less than 10 epochs) usually works well for generalization. See Appendix D of [5] for details. This is consistent with our hypothesis that constrained training should kick in before loss flattening sets in.

3.2. Image Classification for ImageNet

As compared to CIFAR, the ImageNet classification problem [13] is more representative of real world classification problems. We used a variant of the popular ResNet50 [6] model as the classifier. We considered a small (256) and a large (16k) batch size for this experiment, and fixed training budget to be 90 epochs.



(*a*) CIFAR-10 (*b*) CIFAR-100 (*c*) ImageNet

Figure 1: Adam-LAWN vs. Adam (weight decay comprehensively tuned) for a variety of datasets. Adam-LAWN causes little to no drop in generalization performance with increasing batch size.

Method	**CIFAR-10**			**CIFAR-100**			**ImageNet**	
	256	4k	10k	256	4k	10k	256	16k
SGD	**93.99**	93.48	92.99	73.49	71.68	71.07	76.00	74.48
SGD-LAWN	93.96	**93.50**	**93.43**	**73.67**	**72.71**	**71.80**	**76.12**	**75.56**
Adam	93.48	92.93	92.63	70.84	68.91	68.61	71.16	70.60
Adam-LAWN	**93.91**	**93.74**	**93.84**	**72.99**	**73.12**	**72.97**	**76.18**	**76.07**

Table 2: Test Acc. on CIFAR-10, CIFAR-100 and ImageNet. Standard error in the the range of $[0.1, 0.45]$ for CIFAR and $[0.01, 0.08]$ for ImageNet. Details in Appendix D of [5]. LAWN enables Adam to work on image classification tasks with very little drop in performance at large batch sizes. Non-LAWN optimizers have a much steeper drop-off in performance as batch size increases.

Results for batch size 256. Overall results can be found in Table 2. SGD, used in conjunction with momentum and weight decay, has long been the optimizer of choice for image classification. We retain the tuned value for weight decay of the base SGD optimizer for the free phase of SGD-LAWN experiments. SGD-LAWN marginally outperforms SGD.

Adam is well known to perform worse than SGD for image classification tasks [22]. For our experiment, we tuned the learning rate and could only get an accuracy of 71.16%. In comparison, Adam-LAWN achieves an accuracy of more than 76%, marginally surpassing the performance of SGD-LAWN and SGD.

Results for batch size 16k. For the large batch size of 16k, we noticed that LAWN retains strong generalization performance. Both Adam-LAWN and LAMB-LAWN achieve very high accuracy, with Adam-LAWN retaining its performance at such a large batch size by crossing the 76% test accuracy mark. This is with only additonally tuning for the LAWN variants E_{free} and E_{warmup}.

4. Discussion

LAWN is a simple and efficient approach to improving adaptive optimizers such as Adam to match SGD in performance on image classification problems. Given that Adam works better than SGD in other areas such as NLP, LAWN moves Adam closer to being the best off-the-shelf optimizer. Also, the graceful degradation with batch size means that by using parallel/distributed computations training can be sped up a lot.

Finally, the main idea of the paper - that of improving optimizers via layer-wise weight norm constraints - may have implications for deep net training optimization. We started the paper with the note that Adam does aggressive training loss optimization, but loses on performance ultimately. But

this statement may not be true in the constrained phase of LAWN. In this phase it may be true that algorithms with good training also have good performance. This aspect is worth studying in detail.

References

[1] Leon Bottou. Large-scale machine learning with stochastic gradient descent. In *in COMPSTAT*, 2010.

[2] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On Empirical Comparisons of Optimizers for Deep Learning, 2020.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[4] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training imagenet in 1 hour, 2018.

[5] Aman Gupta, Rohan Ramanath, Jun Shi, Anika Ramachandran, Sirou Zhou, Mingzhou Zhou, and S. Sathiya Keerthi. Logit attenuating weight normalization. *CoRR*, abs/2108.05839, 2021. URL https://arxiv.org/abs/2108.05839.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Youngjung Uh, and Jung-Woo Ha. Slowing down the weight norm increase in momentum-based optimizers. *CoRR*, abs/2006.08217, 2020. URL https://arxiv.org/abs/2006.08217.

[8] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. *arXiv preprint arXiv:1803.01814*, 2018.

[9] Nitish Shirish Keskar and Richard Socher. Improving Generalization Performance by Switching from Adam to SGD, 2017.

[10] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[12] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 and CIFAR-100 datasets. 6(1): 1, 2009.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[15] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks, 2020.

[16] Zachary Nado, Justin M Gilmer, Christopher J Shallue, Rohan Anil, and George E Dahl. A large batch optimizer reality check: Traditional, generic optimizers suffice across batch sizes. *arXiv preprint arXiv:2102.06356*, 2021.

[17] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[19] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[22] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning, 2018.

[23] Lei Wu, Chao Ma, and Weinan E. How SGD selects the global minima in over-parameterized learning: A dynamical stability perspective. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, 2018.