Generalization Issues in Adaptive Learning Algorithms

Teja Gupta*
Electrical and Computer Engineering
tejag2@illinois.edu

Akshunna Vaishnav*
Electrical and Computer Engineering
av16@illinois.edu

Vanshika Gupta*
Industrial and Systems Engineering
vg18@illinois.edu

**University of Illinois Urbana Champaign*
Urbana, USA

I. Introduction

The particular problem we are looking at is adaptive learning rate convergence with respect to neural networks. Various algorithms for adaptive learning have been proposed such as Adam and RMSProp. These algorithms are known to have fast training times despite minimal tuning and, as a result, have been widely adopted by the ML community. Despite their popularity, these algorithms have demonstrated issues with generalizability (performance on unseen data). Several cases have been demonstrated where Adam and RMSProp do not converge to the optimal solution. In addition, it has been shown that they have subpar generalization performance compared to Stochastic Gradient Descent (SGD).

In this paper, we have reviewed two approaches to ameliorate these issues. In the proposed solutions, training starts off with an adaptive method such as Adam and transitions into SGD [1], [2], thus, taking advantage of both the fast training time of adaptive methods as well as the generalization performance of SGD.

II. Convergence Issues with Adaptive Algorithms

In lieu of SGD, adaptive learning methods such as Adam and RMSProp have been widely adopted. However, adaptive learning rate algorithms have demonstrated issues with regards to their convergence. They often create models with poor ability to generalize. In other words, the models do not handle unseen data well. While these algorithms demonstrate fast training and high performance on both training and evaluation metrics, their performance plateaus in later stages. There have been several examples in computer vision and natural language processing tasks in which SGD has outperformed adaptive learning algorithms [1]. It has been shown that even for simple quadratic problems, adaptive algorithms such as Adam fail to converge [2].The under-performance of these adaptive algorithms has been attributed to a couple of factors. The first factor is non-uniform scaling of the gradient. The second factor is the presence of an extremely large or small learning rate towards the end of training [1].

The deficiencies noted have been proven for Adam, however these arguments can be easily applied to other adaptive algorithms as well. An experiment conducted to study ResNet-34 on CIFAR-10 found that at the end of training, very small learning rates (less than 0.01) and very large learning rates (greater than 1000) were both present in the model [1], as demonstrated in figure 1.

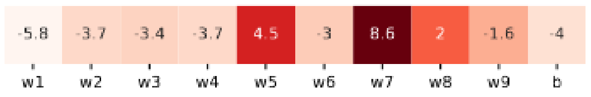

Fig. 1. Learning rates from first nine convolution layers as well as the bias from the linear term. All learning rates are *log* scaled.

A similar analysis was also done with DenseNet on CIFAR-10, demonstrating the inability for Adam to generalize. In addition to regular Adam, clipped versions of Adam such that the learning rates were scaled between 0 and 1 (Adam-Clip(0,1)) and 1 and infinity (Adam-Clip(1,∞)) were also tested [2]. A graph of the training process can be seen in figure 2.

As shown in figure 2, SGD was found to have a testing error of roughly 5% while Adam was found to have a testing error of roughly 7%. Further analysis conducted showed that regardless of the learning rate schedule, SGD outperformed Adam in generalization ability [1]. With regards to Adam-Clip(0,1), no notable difference in performance was found when compared to regular Adam. However, Adam-Clip(1,∞) had a test error of roughly 6%. This corroborates the notion that even small learning rates can have a detrimental impact on generalization ability.

The study also demonstrates mathematically that regardless of the initial learning rate α and parameters β_1 , β_2 in Adam, where $\beta_1 < \sqrt{\beta_2}$, there exist problems in which Adam does not achieve convergence [1].

III. Proposed Solutions

A. Adam with Dynamic Bounds

One approach taken to solve these issues in Adam is the AdaBound algorithm [1]. AdaBound creates dynamic upper and

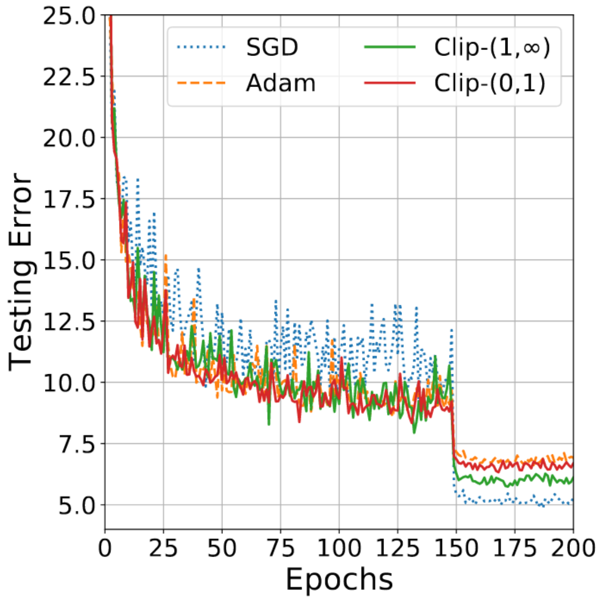

Fig. 2. Training process with SGD, Adam, Adam-Clip(0,1), Adam-Clip(1,∞). As can be shown above there is a generalization gap of 2% between Adam and SGD.

lower bounds on the learning rate during the training process known as η_l and η_u respectively. During the training process, η_l and η_u clip the learning rate, denoted by Clip(η_l,η_u). At the start of the training process, η_l and η_u are initialized to 0 and $infinity$ respectively, denoted as Clip(0,∞). At this point the algorithm behaves identical to Adam. Towards the end of training the η_l and η_u will both converge to the same constant α^*, which is denoted as Clip(α^*,α^*). The algorithm at this point will now act identical to SGD where $\alpha = \alpha^*$. The functions used for the upper and lower bounds were $\eta_l(t) = (1 - \frac{1}{(1-\beta)t+1})\alpha^*$ and $\eta_u(t) = (1 + \frac{1}{(1-\beta)t})\alpha^*$. β is used to control the convergence speed of the bounds and α^* denotes the convergence target [1]. The paper found that overall the choice of parameters did not contribute significantly to the performance of the algorithm and that minimal tuning was required to outperform SGD.

In short, this algorithm will act like the adaptive methods towards the beginning of training and like SGD towards the end. Thus combining the fast training of adaptive methods with the generalization ability of SGD. The paper further demonstrates that there is a upper bound on the regret of the function of $\mathcal{O}(\sqrt{T})$ where T is the number of time steps [1]. Note that the $regret$ of a function is the difference between the total loss and the minimum loss value and is defined as $R_T = \sum t = 1^T f_t(x_t) - min_{x\in F} \sum t = 1^T f_t(x)$ where F is the feasible set and function f is the loss function. A similar algorithm and result was shown for AMSGrad as AMSBound. An outline of the AdaBound algorithm is shown below.

B. *Utilizing SGD in conjunction with Adam*

The next approach to consider, proposed by Keskar and Socher [2], is SWATS (**Sw**itching from **ADAM** **To** **SGD**).

Algorithm 1 AdaBound

Input x ∈ F, initial step α,$\{\beta_{1t}\}_{t=1}^T$, β_2, lower bound function η_l, upper bound function η_u

```
1: $Set\ m_0 = 0,\ v_0 = 0$
2: for $t = 1$ to T do
3:     $g_t = \nabla f_t(x_t)$
4:     $m_t = \beta_{1t} m_{t-1} + (1 - \beta_{1t}) g_t$
5:     $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ and $V_t = diag(v_t)$
6:     $\hat{\eta_t} = Clip(\frac{\alpha}{\sqrt{V_t}}, \eta_l(t), \eta_u(t))$ and $\eta_t = \frac{\hat{\eta_t}}{\sqrt{t}}$
7:     $x_{t+1} = \Pi_{F,diag(\eta_t^{-1})}(x_t - \eta_t) \odot m_t$
8: end for
```

As mentioned in a previous section, Adam outperforms SGD in training as well as initial training generalization. However, SGD still outclasses Adam in post-training generalization. As such, using Adam when when starting out and then switch to SGD as needed seems like a plausible approach. To compare this new approach (SWATS) with Adam effectively, care was taken to not add any additional hyperparameters.

This approach differs from the previously mentioned approach due to two key factors: first, there are no bounds added that allow for a second level convergence. Similar to the previous approach, AMSGrad was also found to be inadequate in areas where Adam failed to converge [1]. Hence, the idea of a ”switchover point” is used. This *switchover* point, calculated during the training phase, determines when Adam is switched over to SGD. There are two main questions to consider here: (1) What is the ideal time to switch over from Adam such that the benefits of inital progress are kept, and (2) What should be the learning rate when Adam is switched to SGD. Focusing on the first question, it was found that switching too late does not yield generalization improvements, and switching too early causes the optimizer to not benefit from the rapid initial progress of Adam. Figure 3 highlights a few tests done to determine which switchover point is ideal.

As we can see in this figure, switching early allows the model to achieve SGD-like testing accuracy, but switching too late leads to the previously mentioned generalization gap akin to Adam.

Given an iterate w_k, stochastic gradient g_k, and step p_k according to Adam, assume that $p_k \neq 0$ and $p_k^T g_k < 0$, which are common assumptions in order to arrive at convergence.

Given that no first-order exponential averaging is used in Adam, we know that the next iterate can be found through the equation

$$w_{k+1} = w_k + p_k.$$

The method that this paper then uses to determine a feasible learning rate for SGD, γ_k, is through solving the following subproblem:

$$\text{proj}_{-\gamma_k g k} p_k = p_k.$$

Here, proj$_a b$ is the orthogonal projection of a onto b, as usual. The closed form solution was found to be

$$\gamma_k = \frac{p_k^T p_k}{-p_k^T g_k}.$$

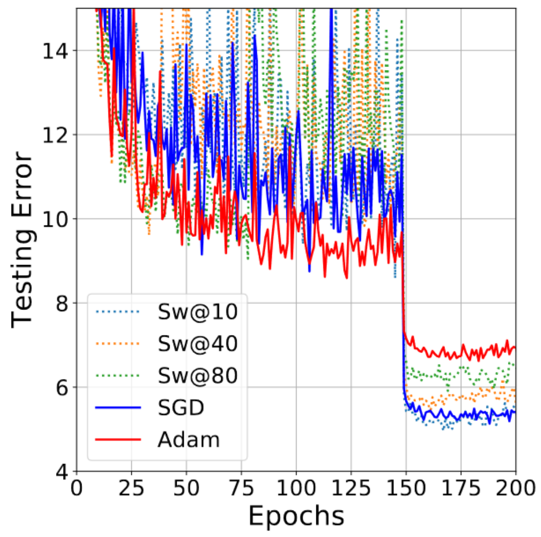

Fig. 3. Training the DenseNet architecture on the CIFAR-10 data set using Adam and switching to SGD with learning rate with learning rate 0.1 and momentum 0.9 after (10,40,80) epochs. Here, Sw@# denotes the switchover point given # number of epochs.

There is also a second value that is kept track of, λ_k, which is an exponential average that depends on the β_2 constant of Adam (this constant was introduced in the discussion of the previous paper) as follows:

$$\lambda_k = \beta_2 \lambda_{k-1} + (1 - \beta_2)\gamma_k$$

This parameter makes use of a pre-existing parameter, bypassing the need for additional hyperparameters. This will be used in finding the switchover point.

The final question to answer is how to decide when to switch from Adam to SGD. For this, a simple criterion was suggested:

$$|\frac{\lambda_k}{1 - \beta_2^k} - \gamma_k| < \epsilon$$

Note two things here: first, that this condition needs to be checked at every iteration for $k > 1$ and second, that the bias correction of the λ_k term is done by dividing it by $1 - \beta_2^k$, and this is necessary to prevent the influence of the initialization to zero during the start of the training. As soon as this condition is true, Adam is switched to SGD with learning rate Λ as follows:

$$\Lambda := \frac{\lambda_k}{1 - \beta_2^k}$$

Summarizing the previous two sections, the overall algorithm can be seen in the "Algorithm 2" overview picture.

Algorithm 2 SWATS

Inputs: Objective function f, initial point w_0, learning rate $\alpha = 10^{-3}$, accumulator coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-9}$, phase=Adam.

```
 1: Initialize $k \leftarrow 0$, $m_k \leftarrow 0$, $a_k \leftarrow 0$, $\lambda_k \leftarrow 0$
 2: while stopping criterion not met do
 3:     $k = k + 1$
 4:     Compute stochastic gradient $g_k = \hat{\nabla} f(w_{k-1})$
 5:     if phase = SGD then
 6:         $v_k = \beta_1 v_{k-1} + g_k$
 7:         $w_k = w_{k-1} - (1 - \beta_1)\Lambda v_k$
 8:         continue
 9:     end if
10:     $m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k$
11:     $a_k = \beta_2 a_{k-1} + (1 - \beta_2) g_k^2$
12:     $p_k = -\alpha_k \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{m_k}{\sqrt{a_k}+\epsilon}$
13:     $w_k = w_k + p_k$
14:     if $p_k^T g_k \neq 0$ then
15:         $\gamma_k = \frac{p_k^T p_k}{-p_k^T g_k}$
16:         $\lambda_k = \beta_2 \lambda_{k-1} + (1 - \beta_2)\gamma_k$
17:         if $k > 1$ and $|\frac{\lambda_k}{(1-\beta_2^k)} - \gamma_k| < \epsilon$ then
18:             phase = SGD
19:             $v_k = 0$
20:             $\Lambda = \lambda_k / (1 - \beta_2^k)$
21:         end if
22:     else
23:         $\lambda_k = \lambda_{k-1}$
24:     end if
25: end while
return $w_k$
```

Note: The text in blue represents overlaps with the Adam algorithm.

IV. Experiments

A. Experiments with AdaBound

The performance of the proposed algorithm has been compared with the following optimization methods: SGD(M), AdaGrad, Adam, and AMSGrad. Three classification tasks have been used, namely the MNIST image classification task [3], the CIFAR-10 image classification task [4] , and the language modeling task on Penn Treebank [5] as shown in Figure 4.

Hyperparameters corresponding to each optimization method are chosen using logarithmically spaced grid. For e.g. for SGD, α values are first coarsely selected from a grid of $\{100, 10, 1, 0.1, 0.01\}$ and then fine tuned on the obtained interval. Batch size, dropout probability, weight decay are chosen based on the base architecture recommendations.

For the feedfoward neural network on MNIST dataset, training accuracy converges to 100% for all methods in 100 epochs. However, in testing, the proposed methods display slight improvement over the original AdaGrad and Adam, while SGD performs the best.

For Convolutional Neural Network DenseNet on CIFAR-10 dataset, the proposed methods outperform their prototypes by $\sim 2\%$ and SGD by very slight margin on the test data at the end of training. With ResNet, the AdaBound method even outperforms SGD by $\sim 1\%$. Hence, the slackness in

Dataset	Network Type	Architecture
MNIST	Feedforward	1-Layer Perceptron
CIFAR-10	Deep Convolutional	DenseNet-121
CIFAR-10	Deep Convolutional	ResNet-34
Penn Treebank	Recurrent	1-Layer LSTM
Penn Treebank	Recurrent	2-Layer LSTM
Penn Treebank	Recurrent	3-Layer LSTM

Fig. 4. Models Used

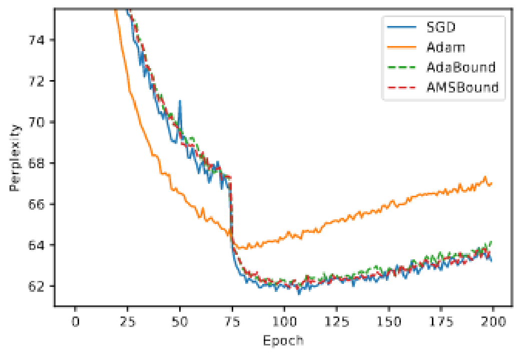

Fig. 5. Perplexity curves on the test set for 3-Layer (L3) LSTM

generalization ability of Adaptive Methods is improved by the bounds used in the proposed methods.

For the recurrent neural network on Penn Treebank, experiments are conducted using LSTM model with 1, 2 and 3 layers. With increasing layers, Adam's performance degrades and proposed methods outperform by greater margin: $\sim 2.8\%$ with $L3$ and $\sim 1.1\%$ in $L1$. The AdaBound methods perform similar to SGD in terms of the achieved accuracy but at the same time have smoother performance curves as shown in Figure 5.

Further, the method is also tested with popular computer vision tasks and it outperforms Adam and AMSGrad with a faster rate of convergence and also achieves a higher test accuracy at the end of training. As can be seen, experiments discussed above involve models with increasing complexities from Perceptron to RNN. Since the existence of extreme learning rates is more prominent in complex models, Bound methods (AdaBound and AMSBound) demonstrate an increasing margin of improvement with respect to the original methods in more complicated models.

B. Experiments with SWATS

The proposed strategy is compared with Adam and SGD. Image Classification is performed using ResNet-32, DenseNet, PyramidNet and SENet on the CIFAR-10 and CIFAR-100 data sets. Language modelling is performed on Penn Treebank (PTB) and WikiText-2 (WT-2) data sets using the AWD-LSTM and AWD-QRNN [6] architectures.

The gradients are clipped to a norm of 0.25 in SGD, but not for Adam and SWATS as performance is observed to enhance this way. The learning rates

Model	Data Set	SGDM	Adam	SWATS	Λ	Switchover Point (epochs)
ResNet-32	CIFAR-10	0.1	0.001	0.001	0.52	1.37
DenseNet	CIFAR-10	0.1	0.001	0.001	0.79	11.54
PyramidNet	CIFAR-10	0.1	0.001	0.0007	0.85	4.94
SENet	CIFAR-10	0.1	0.001	0.001	0.54	24.19
ResNet-32	CIFAR-100	0.3	0.002	0.002	1.22	10.42
DenseNet	CIFAR-100	0.1	0.001	0.001	0.51	11.81
PyramidNet	CIFAR-100	0.1	0.001	0.001	0.76	18.54
SENet	CIFAR-100	0.1	0.001	0.001	1.39	2.04
LSTM	PTB	$55^\dagger$	0.003	0.003	7.52	186.03
QRNN	PTB	$35^\dagger$	0.002	0.002	4.61	184.14
LSTM	WT-2	$60^\dagger$	0.003	0.003	1.11	259.47
QRNN	WT-2	$60^\dagger$	0.003	0.004	14.46	295.71
ResNet-18	Tiny-ImageNet	0.2	0.001	0.0007	1.71	48.91

Fig. 6. Optimal hyperparameters for SGD(M), Adam and SWATS for all experiments

of Adam and SWATS are chosen from a grid of $\{0.0005, 0.0007, 0.001, 0.002, 0.003, 0.004, 0.005\}$ while β values are fixed as per model recommendations. Batch size, dropout probability, l_2-norm decay etc. are based on recommendations of the respective base architectures. 300 epochs are used for training and the learning rate is reduced by 10 on epoch 150, 225 and 262.

Across all architectures/data sets in image classification, Adam fails to generalize well and its performance deteriorates by the end of training as compared to others. SWATS' performance is similar to that of SGD in most cases even though the error in SWATS increases momentarily when the switch from Adam to SGD occurs (see Figure 6).

For language modelling task, Adam outperforms all and converges faster to a higher accuracy as compared to SGD, whereas SWATS is able to achieve similar generalization as Adam. Overall, it can be seen that SWATS performs almost as good as the best of SGD and Adam.

V. Further Work

Despite the proposed solutions, there still remains ample scope for work that needs done, alongside many unresolved questions. It is still not entirely clear why SGD does so well when compared to adaptive algorithms. While it is clear that starting off with an adaptive learning algorithm, such as Adam, and transitioning to SGD has brought improvements in terms of solving the generalization problem, it is still debatable as to what is the best way to go about this transition. Additionally, studies have been ongoing as to how can we do better in terms of improving the generalization ability of models.

References

[1] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," *arXiv preprint arXiv:1902.09843*, 2019.
[2] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," 2017.
[3] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.
[4] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[5] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," 1993.
[6] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," *arXiv preprint arXiv:1611.01576*, 2016.