Improving Generalization Performance by Switching from Adam to SGD

Nitish Shirish Keskar [1] **Richard Socher** [1]

Abstract

Despite superior training outcomes, adaptive optimization methods such as Adam, Adagrad or RMSprop have been found to generalize poorly compared to Stochastic gradient descent (SGD). These methods tend to perform well in the initial portion of training but are outperformed by SGD at later stages of training. We investigate a hybrid strategy that begins training with an adaptive method and switches to SGD when appropriate. Concretely, we propose SWATS, a simple strategy which **Sw**itches from **A**dam **t**o **S**GD when a triggering condition is satisfied. The condition we propose relates to the projection of Adam steps on the gradient subspace. By design, the monitoring process for this condition adds very little overhead and does not increase the number of hyperparameters in the optimizer. We report experiments on several standard benchmarks such as: ResNet, SENet, DenseNet and PyramidNet for the CIFAR-10 and CIFAR-100 data sets, ResNet on the tiny-ImageNet data set and language modeling with recurrent networks on the PTB and WT2 data sets. The results show that our strategy is capable of closing the generalization gap between SGD and Adam on a majority of the tasks.

1. Introduction

Stochastic gradient descent (SGD) (Robbins & Monro, 1951) has emerged as one of the most used training algorithms for deep neural networks. Despite its simplicity, SGD performs well empirically across a variety of applications but also has strong theoretical foundations. This includes, but is not limited to, guarantees of saddle point avoidance (Lee et al., 2016), improved generalization (Hardt et al., 2015; Wilson et al., 2017) and interpretations as Bayesian inference (Mandt et al., 2017).

Training neural networks is equivalent to solving the following non-convex optimization problem,

$$\min_{w \in \mathbb{R}^n} \quad f(w),$$

where f is a loss function. The iterations of SGD can be described as:

$$w_k = w_{k-1} - \alpha_{k-1} \hat{\nabla} f(w_{k-1}),$$

where w_k denotes the k^{th} iterate, α_k is a (tuned) step size sequence, also called the learning rate, and $\hat{\nabla} f(w_k)$ denotes the stochastic gradient computed at w_k. A variant of SGD (SGDM), that uses the inertia of the iterates to accelerate the training process, has also found to be successful in practice (Sutskever et al., 2013). The iterations of SGDM can be described as:

$$v_k = \beta v_{k-1} + \hat{\nabla} f(w_{k-1})$$
$$w_k = w_{k-1} - \alpha_{k-1} v_k,$$

where $\beta \in [0, 1)$ is a momentum parameter and v_0 is initialized to 0.

One disadvantage of SGD is that it scales the gradient uniformly in all directions; this can be particularly detrimental for ill-scaled problems. This also makes the process of tuning the learning rate α circumstantially laborious.

To correct for these shortcomings, several *adaptive* methods have been proposed which diagonally scale the gradient via estimates of the function's curvature. Examples of such methods include Adam (Kingma & Ba, 2015), Adagrad (Duchi et al., 2011) and RMSprop (Tieleman & Hinton, 2012). These methods can be interpreted as methods that use a vector of learning rates, one for each parameter, that are adapted as the training algorithm progresses. This is in contrast to SGD and SGDM which use a scalar learning rate uniformly for all parameters.

Adagrad takes steps of the form

$$w_k = w_{k-1} - \alpha_{k-1} \frac{\hat{\nabla} f(w_{k-1})}{\sqrt{v_{k-1}} + \epsilon}, \quad \text{where} \qquad (1)$$
$$v_{k-1} = \sum_{j=1}^{k-1} \hat{\nabla} f(w_j)^2.$$

[1]Salesforce Research, Palo Alto, CA – 94301. Correspondence to: Nitish Shirish Keskar <nkeskar@salesforce.com>.

RMSProp uses the same update rule as (1), but instead of accumulating v_k in a monotonically increasing fashion, uses an RMS-based approximation instead, i.e.,

$$v_{k-1} = \beta v_{k-2} + (1-\beta)\hat{\nabla} f(w_{k-1})^2.$$

In both Adagrad and RMSProp, the accumulator v is initialized to 0. Owing to the fact that v_k is monotonically increasing in each dimension for Adagrad, the scaling factor for $\hat{\nabla} f(w_{k-1})$ monotonically decreases leading to slow progress. RMSProp corrects for this behavior by employing an *average* scale instead of a *cumulative* scale. However, because v is initialized to 0, the initial updates tend to be noisy given that the scaling estimate is biased by its initialization. This behavior is rectified in Adam by employing a bias correction. Further, it uses an exponential moving average for the step in lieu of the gradient. Mathematically, the Adam update equation can be represented as:

$$w_k = w_{k-1} - \alpha_{k-1} \cdot \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \cdot \frac{m_{k-1}}{\sqrt{v_{k-1}} + \epsilon}, \quad \text{where} \tag{2}$$

$$m_{k-1} = \beta_1 m_{k-2} + (1-\beta_1)\hat{\nabla} f(w_{k-1}),$$
$$v_{k-1} = \beta_2 v_{k-2} + (1-\beta_2)\hat{\nabla} f(w_{k-1})^2. \tag{3}$$

Adam has been used in many applications owing to its competitive performance and its ability to work well despite minimal tuning (Karpathy, 2017). Recent work, however, highlights the possible inability of adaptive methods to perform on par with SGD when measured by their ability to generalize (Wilson et al., 2017).

Furthermore, the authors also show that for even simple quadratic problems, adaptive methods find solutions that can be orders-of-magnitude worse at generalization than those found by SGD(M).

Indeed, for several state-of-the-art results in language modeling and computer vision, the optimizer of choice is SGD (Merity et al., 2017; Loshchilov & Hutter, 2016; He et al., 2015). Interestingly however, in these and other instances, Adam outperforms SGD in both training and generalization metrics in the initial portion of the training, but then the performance stagnates. This motivates the investigation of a strategy that combines the benefits of Adam, viz. good performance with default hyperparameters and fast initial progress, and the generalization properties of SGD. Given the insights of Wilson et al. (2017) which suggest that the lack of generalization performance of adaptive methods stems from the non-uniform scaling of the gradient, a natural *hybrid* strategy would begin the training process with Adam and switch to SGD when appropriate. To investigate this further, we propose SWATS, a simple strategy that combines the best of both worlds by **S**witching from **A**dam **t**o **S**GD. The switch is designed to be automatic and one that does not introduce any more hyper-parameters. The choice of not adding additional hyperparameters is deliberate since it allows for a fair comparison between Adam and SWATS. Our experiments on several architectures and data sets suggest that such a strategy is indeed effective.

Several attempts have been made at improving the convergence and generalization performance of Adam. The closest to our proposed approach is (Zhang et al., 2017) in which the authors propose ND-Adam, a variant of Adam which preserves the gradient direction by a nested optimization procedure. This, however, introduces an additional hyperparameter along with the $(\alpha, \beta_1, \beta_2)$ used in Adam. Further, empirically, this adaptation sacrifices the rapid initial progress typically observed for Adam. In Anonymous (2018), the authors investigate Adam and ascribe the poor generalization performance to training issues arising from the non-monotonic nature of the steps. The authors propose a variant of Adam called AMSGrad which monotonically reduces the step sizes and possesses theoretical convergence guarantees. Despite these guarantees, we empirically found the generalization performance of AMSGrad to be similar to that of Adam on problems where a generalization gap exists between Adam and SGD. We note that in the context of the hypothesis of Wilson et al. (2017), all of the aforementioned methods would still yield poor generalization given that the scaling of the gradient is non-uniform.

The idea of switching from an adaptive method to SGD is not novel and has been explored previously in the context of machine translation (Wu et al., 2016) and ImageNet training (Akiba et al., 2017). Wu et al. (2016) use such a mixed strategy for training and tune both the switchover point and the learning rate for SGD after the switch. Akiba et al. (2017) use a similar strategy but use a convex combination of RMSProp and SGD steps whose contributions and learning rates are tuned.

In our strategy, the switchover point and the SGD learning rate are both learned as a part of the training process. We monitor a projection of the Adam step on the gradient subspace and use its exponential average as an estimate for the SGD learning rate after the switchover. Further, the switchover is triggered when no change in this monitored quantity is detected. We describe this strategy in detail in Section 2. In Section 3, we describe our experiments comparing Adam, SGD and SWATS on a host of benchmark problems. Finally, in Section 4, we present ideas for future research and concluding remarks. We conclude this section by emphasizing the goal of this work is less to propose a new training algorithm but rather to empirically investigate the viability of hybrid training for improving generalization.

2. SWATS

To investigate the generalization gap between Adam and SGD, let us consider the training of the CIFAR-10 data set (Krizhevsky & Hinton, 2009) on the DenseNet architecture (Iandola et al., 2014). This is an example of an instance where a significant generalization gap exists between Adam and SGD. We plot the performance of Adam and SGD on this task but also consider a variant of Adam which we call Adam-Clip(p, q). Given (p, q) such that $p < q$, the iterates for this variant take on the form

$$w_k = w_{k-1} - \text{clip}\left(\frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k}\frac{\alpha_{k-1}}{\sqrt{v_{k-1}}+\epsilon}, p\cdot\alpha_{sgd}, q\cdot\alpha_{sgd}\right) m_{k-1}.$$

Here, α_{sgd} is the tuned value of the learning rate for SGD that leads to the best performance for the same task. The function clip(x, a, b) clips the vector x element-wise such that the output is constrained to be in $[a, b]$. Note that Adam-Clip$(1, 1)$ would correspond to SGD. The network is trained using Adam, SGD and two variants: Adam-Clip$(1, \infty)$, Adam-Clip$(0, 1)$ with tuned learning rates for 200 epochs, reducing the learning rate by 10 after 150 epochs. The goal of this experiment is to investigate the effect of constraining the large and small step sizes that Adam implicitly learns, i.e., $\frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k}\frac{\alpha_{k-1}}{\sqrt{v_{k-1}}+\epsilon}$, on the generalization performance of the network. We present the results in Figure 1.

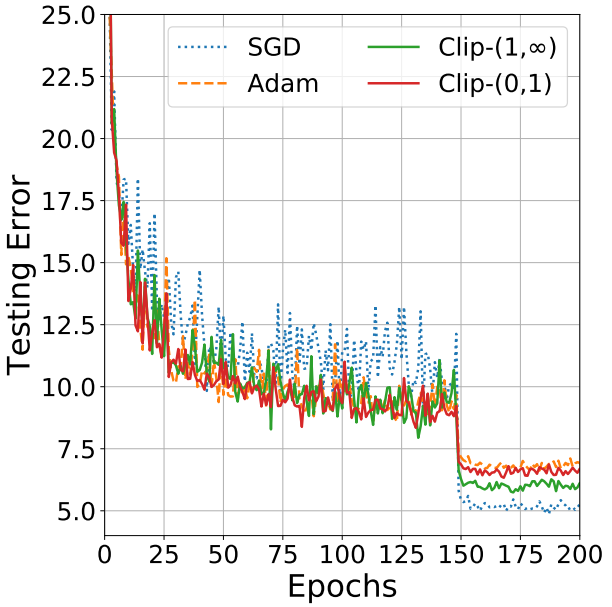


Figure 1. Training the DenseNet architecture on the CIFAR-10 data set with four optimizers: SGD, Adam, Adam-Clip$(1, \infty)$ and Adam-Clip$(0, 1)$. SGD achieves the best testing accuracy while training with Adam leads to a generalization gap of roughly 2%. Setting a minimum learning rate for each parameter of Adam partially closes the generalization gap.

As seen from Figure 1, SGD converges to the expected testing error of $\approx 5\%$ while Adam stagnates in performance at around $\approx 7\%$ error. We note that fine-tuning of the learning rate schedule (primarily the initial value, reduction amount and the timing) did not lead to better performance. Also, note that the rapid initial progress of Adam relative to SGD. This experiment is in agreement with the experimental observations of Wilson et al. (2017). Interestingly, Adam-Clip$(0, 1)$ has no tangible effect on the final generalization performance while Adam-Clip$(1, \infty)$ partially closes the generalization gap by achieving a final accuracy of $\approx 6\%$. We observe similar results for several architectures, data sets and modalities whenever a generalization gap exists between SGD and Adam. This stands as evidence that the step sizes learned by Adam could circumstantially be too small for effective convergence. This observation regarding the need to lower-bound the step sizes of Adam is similar to the one made in Anonymous (2018), where the authors devise a one-dimensional example in which infrequent but large gradients are not emphasized sufficiently causing the non-convergence of Adam.

Given the potential insufficiency of Adam, even when constraining one side of the accumulator, we consider *switching* to SGD once we have reaped the benefits of Adam's rapid initial progress. This raises two questions: (a) when to switch over from Adam to SGD, and (b) what learning rate to use for SGD after the switch. Assuming that the learning rate of SGD after the switchover is tuned, we found that switching too late does not yield generalization improvements while switching too early may cause the hybrid optimizer to not benefit from Adam's initial progress. Indeed, as shown in Figure 2, switching after 10 epochs leads to a learning curve very similar to that of SGD, while switching after 80 epochs leads to inferior testing accuracy of $\approx 6.5\%$. To investigate the efficacy of a hybrid strategy whilst ensuring no increase in the number of hyperparameters (a necessity for fair comparison with Adam), we propose SWATS, a strategy that automates the process of switching over by determining both the switchover point and the learning rate of SGD after the switch.

2.1. Learning rate for SGD after the switch

Consider an iterate w_k with a stochastic gradient g_k and a step computed by Adam, p_k. For the sake of simplicity, assume that $p_k \neq 0$ and $p_k^T g_k < 0$. This is a common requirement imposed on directions to derive convergence (Nocedal & Wright, 2006). In the case when $\beta_1 = 0$ for Adam, i.e., no first-order exponential averaging is used, this

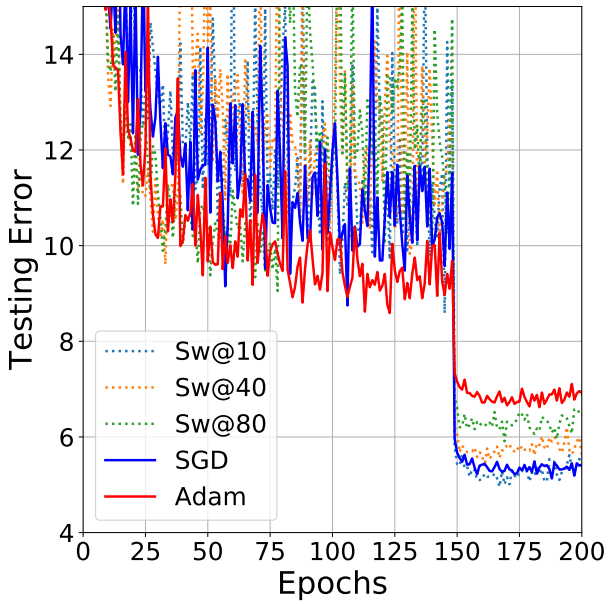


Figure 2. Training the DenseNet architecture on the CIFAR-10 data set using Adam and switching to SGD with learning rate with learning rate 0.1 and momentum 0.9 after $(10, 40, 80)$ epochs; the switchover point is denoted by Sw@ in the figure. Switching early enables the model to achieve testing accuracy comparable to SGD but switching too late in the training process leads to a generalization gap similar to Adam.

is trivially true since

$$p_k = -\underbrace{\frac{\sqrt{1-\beta_2^{k+1}}}{1-\beta_1^{k+1}}\frac{\alpha_k}{\sqrt{v_k}+\epsilon}}_{:=\text{diag}(H_k)} g_k, t$$

with $H_k \succ 0$ where $\text{diag}(A)$ denotes the vector constructed from the diagonal of A. Ordinarily, to train using Adam, we would update the iterate as:

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

To determine a feasible learning rate for SGD, γ_k, we propose solving the subproblem for finding γ_k

$$\text{proj}_{-\gamma_k g_k} p_k = p_k$$

where $\text{proj}_a b$ denotes the orthogonal projection of a onto b. This scalar optimization problem can be solved in closed form to yield:

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

since

$$p_k = \text{proj}_{-\gamma_k g_k} p_k = -\gamma_k \frac{g_k^T p_k}{p_k^T p_k} p_k$$

implies the above equality.

Geometrically, this can be interpreted as the scaling necessary for the gradient that leads to its projection on the Adam step p_k to be p_k itself; see Figure 3. Note that this is not the same as an orthogonal projection of p_k on $-g_k$. Empirically, we found that an orthogonal projection consistently underestimates the SGD learning rate necessary, leading to much smaller SGD steps. Indeed, the ℓ_2 norm of an orthogonally projected step will always be lesser than or equal to that of p_k, which is undesirable given our needs. The non-orthogonal projection proposed above does not suffer from this problem, and empirically we found that it estimates the SGD learning rate well. A simple scaling rule of $\gamma_k = \frac{\|p\|}{\|g\|}$ was also not found to be successful. We attribute this to the fact that a scaling rule of this form ignores the relative importance of the coordinate directions and tends to amplify the importance of directions with a large step p but small first-order importance g, and vice versa.

Note again that if no momentum ($\beta_1 = 0$) is employed in Adam, then necessarily $\gamma_k > 0$ since $H_k \succ 0$. We should mention in passing that in this case γ_k is equivalent to the reciprocal of the Rayleigh Quotient of H_k^{-1} with respect to the vector p_k.

Since γ_k is a noisy estimate of the scaling needed, we maintain an exponential average initialized at 0, denoted by λ_k such that

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

We use β_2 of Adam, see (3), as the averaging coefficient since this reuse avoids another hyperparameter and also because the performance is relatively invariant to fine-grained specification of this parameter.

2.2. Switchover Point

Having answered the question of what learning rate λ_k to choose for SGD after the switch, we now discuss when to switch. We propose checking a simple, yet powerful, criterion:

$$\left|\frac{\lambda_k}{1-\beta_2^k} - \gamma_k\right| < \epsilon, \tag{4}$$

at every iteration with $k > 1$. The condition compares the bias-corrected exponential averaged value and the current value (γ_k). The bias correction is necessary to prevent the influence of the zero initialization during the initial portion of training. Once this condition is true, we switch over to SGD with learning rate $\Lambda := \frac{\lambda_k}{(1-\beta_2^k)}$. We also experimented with more complex criteria including those involving monitoring of gradient norms. However, we found that this simple un-normalized criterion works well across a variety of different applications.

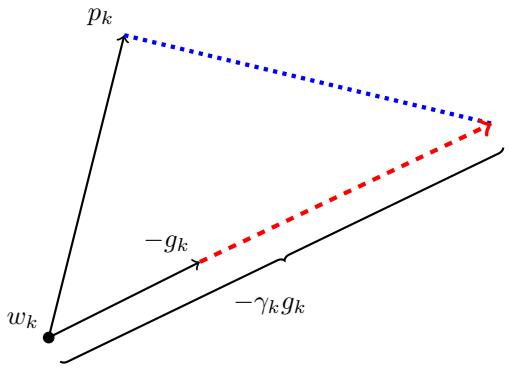


Figure 3. Illustrating the learning rate for SGD (γ_k) estimated by our proposed projection given an iterate w_k, a stochastic gradient g_k and the Adam step p_k.

In the case when $\beta_1 > 0$, we switch to SGDM with learning rate $(1-\beta_1)\Lambda$ and momentum parameter β_1. The $(1-\beta_1)$ factor is the common momentum correction. Refer to Algorithm 1 for a unified view of the algorithm. The text in blue denotes operations that are also present in Adam.

3. Numerical Results

To demonstrate the efficacy of our approach, we present numerical experiments comparing the proposed strategy with Adam and SGD. We consider the problems of image classification and language modeling.

For the former, we experiment with four architectures: ResNet-32 (He et al., 2015), DenseNet (Iandola et al., 2014), PyramidNet (Han et al., 2016), and SENet (Hu et al., 2017) on the CIFAR-10 and CIFAR-100 data sets (Krizhevsky & Hinton, 2009). The goal is to classify images into one of 10 classes for CIFAR-10 and 100 classes for CIFAR-100. The data sets contain 50000 32×32 RGB images in the training set and 10000 images in the testing set. We choose these architectures given their superior performance on several image classification benchmarking tasks. For a large-scale image classification experiment, we experiment with the Tiny-ImageNet data set[1] on the ResNet-18 architecture (He et al., 2015). This data set is a subset of the ILSVRC 2012 data set (Deng et al., 2009) and contains 200 classes with 500 224×224 RGB images per class in the training set and 50 per class in the validation and testing sets. We choose this data set given that it is a good proxy for the performance on the larger ImageNet data set.

We also present results for word-level language modeling where the task is to take as inputs a sequence of words and predict the next word. We choose this task because of its broad importance, the inherent difficulties that arise due to long term dependencies (Hochreiter & Schmidhuber, 1997), and since it is a proxy for other sequence learning tasks such as machine translation (Bahdanau et al., 2014). We use the Penn Treebank (PTB) (Mikolov et al., 2011) and the larger WikiText-2 (WT-2) (Merity et al., 2016) data sets and experimented with the AWD-LSTM and AWD-QRNN architectures. In the case of SGD, we clip the gradients to a norm of 0.25 while we perform no such clipping for Adam and SWATS. We found that the performance of SGD deteriorates without clipping and that of Adam and SWATS with. The AWD-LSTM architecture uses a multi-layered LSTM network with learned embeddings while the AWD-QRNN replaces the expensive LSTM layer by the cheaper QRNN layer (Bradbury et al., 2016) which uses convolutions instead of recurrences. The model is regularized with DropConnect (Wan et al., 2013) on the hidden-to-hidden connections as well as other strategies such as weight decay, embedding-softmax weight tying, activity regularization and temporal activity regularization. We refer the reader

[1] https://tiny-imagenet.herokuapp.com/

Algorithm 1 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.

Initialize $k \leftarrow 0$, $m_k \leftarrow 0$, $a_k \leftarrow 0$, $\lambda_k \leftarrow 0$
while stopping criterion not met **do**
  $k = k + 1$
  Compute stochastic gradient $g_k = \hat{\nabla} f(w_{k-1})$
  if phase = SGD **then**
    $v_k = \beta_1 v_{k-1} + g_k$
    $w_k = w_{k-1} - (1-\beta_1)\Lambda v_k$
    continue
  end if
  $m_k = \beta_1 m_{k-1} + (1-\beta_1) g_k$
  $a_k = \beta_2 a_{k-1} + (1-\beta_2) g_k^2$
  $p_k = -\alpha_k \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{m_k}{\sqrt{a_k}+\epsilon}$
  $w_k = w_k + p_k$
  if $p_k^T g_k \neq 0$ **then**
    $\gamma_k = \frac{p_k^T p_k}{-p_k^T g_k}$
    $\lambda_k = \beta_2 \lambda_{k-1} + (1-\beta_2)\gamma_k$
    if $k > 1$ and $|\frac{\lambda_k}{(1-\beta_2^k)} - \gamma_k| < \epsilon$ **then**
      phase = SGD
      $v_k = 0$
      $\Lambda = \lambda_k / (1-\beta_2^k)$
    end if
  else
    $\lambda_k = \lambda_{k-1}$
  end if
end while
return w_k

to (Merity et al., 2016) for additional details regarding the data sets including the sizes of the training, validation and testing sets, size of the vocabulary, and source of the data.

For our experiments, we tuned the learning rate of all optimizers, and report the best-performing configuration in terms of generalization. The learning rate of Adam and SWATS were chosen from a grid of $\{0.0005, 0.0007, 0.001, 0.002, 0.003, 0.004, 0.005\}$. For both optimizers, we use the (default) recommended values $(\beta_1, \beta_2) = (0.9, 0.999)$. Note that this implies that, in all cases, we switch from Adam to SGDM with a momentum coefficient of 0.9. For tuning the learning rate for the SGD(M) optimizer, we first coarsely tune the learning rate on a logarithmic scale from 10^{-3} to 10^{2} and then fine-tune the learning rate. For all cases, we experiment with and without employing momentum but don't tune this parameter ($\beta = 0.9$). We found this overall procedure to perform better than a generic grid-search or hyperparameter optimization given the vastly different scales of learning rates needed for different modalities. For instance, SGD with learning rate 0.7 performed best for the DenseNet task on CIFAR-10 but for the PTB language modeling task using the LSTM architecture, a learning rate of 50 for SGD was necessary. Hyperparameters such as batch size, dropout probability, ℓ_2-norm decay etc. were chosen to match the recommendations of the respective base architectures. We trained all networks for a total of 300 epochs and reduced the learning rate by 10 on epochs 150, 225 and 262. This scheme was surprisingly powerful at obtaining good performance across the different modalities and architectures. The experiments were coded in PyTorch[2] and conducted using job scheduling on 16 NVIDIA Tesla K80 GPUs for roughly 3 weeks.

The experiments comparing SGD, Adam and SWATS on the CIFAR and Tiny-ImageNet data sets are presented in Figures 4 and 5, respectively. The experiments comparing the optimizers on the language modeling tasks are presented in Figure 6. In Table 1, we summarize the meta-data concerning our experiments including the learning rates that achieved the best performance, and, in the case of SWATS, the number of epochs before the switch occurred and the learning rate (Λ) for SGD after the switch. Finally, in Figure 7, we depict the evolution of the estimated SGD learning rate (γ_k) as the algorithm progresses on two representative tasks.

With respect to the image classification data sets, it is evident that, across different architectures, on all three data sets, Adam fails to find solutions that generalize well despite making good initial progress. This is in agreement with the findings of (Wilson et al., 2017). As can be seen from Table 1, the switch from Adam to SGD happens within the first 20 epochs for most CIFAR data sets and at epoch 49 for Tiny-ImageNet. Curiously, in the case of the Tiny-ImageNet problem, the switch from Adam to SGD leads to significant but temporary degradation in performance. Despite the testing accuracy dropping from 80% to 52% immediately after the switch, the model recovers and achieves a better peak testing accuracy compared to Adam. We observed similar outcomes for several other architectures on this data set.

In the language modeling tasks, Adam outperforms SGD not only in final generalization performance but also in the number of epochs necessary to attain that performance. This is not entirely surprising given that Merity et al. (2017) required iterate averaging for SGD to achieve state-of-the-art performance despite gradient clipping or learning rate decay rules. In this case, SWATS switches over to SGD, albeit later in the training process, but achieves comparable generalization performance to Adam as measured by the lowest validation perplexity achieved in the experiment. Again, as in the case of the Tiny-ImageNet experiment (Figure 5), the switch may cause a temporary degradation in performance from which the model is able to recover.

These experiments suggest that it is indeed possible to combine the best of both worlds for these tasks: in all the tasks described, SWATS performs almost as well as the best algorithm amongst SGD and Adam, and in several cases achieves a good initial decrease in the error metric.

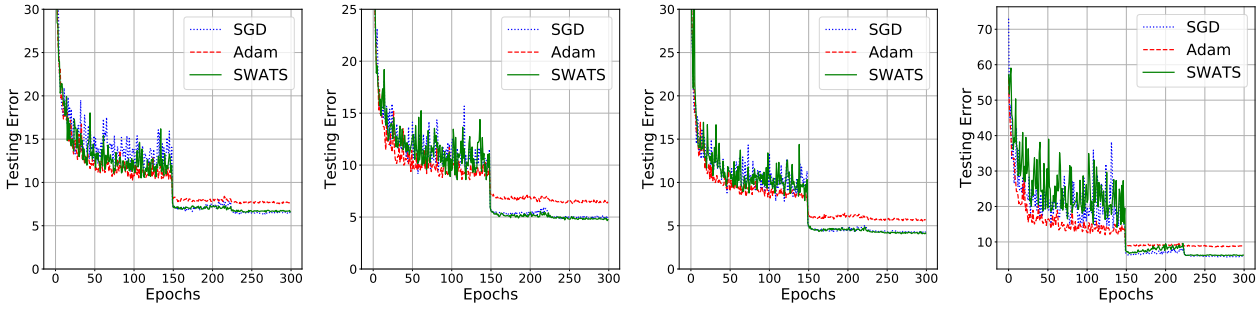
Figure 7 shows that the estimated learning rate for SGD (γ_k) is noisy but convergent (in mean), and that it converges to a value of similar scale as the value obtained by tuning the SGD optimizer (see Table 1). We emphasize that other than the learning rate, no other hyperparameters were tuned between the experiments.

4. Discussion and Conclusion

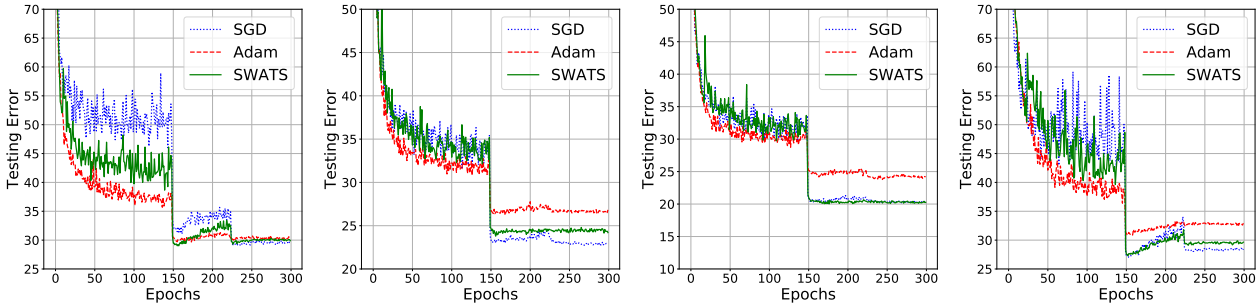
Wilson et al. (2017) pointed to the insufficiency of adaptive methods, such as Adam, Adagrad and RMSProp, at generalizing in a fashion comparable to that of SGD. In the case of a convex quadratic function, the authors demonstrate that adaptive methods provably converge to a point with orders-of-magnitude worse generalization performance than SGD. The authors attribute this generalization gap to the scaling of the per-variable learning rates definitive of adaptive methods as we explain below.

Nevertheless, adaptive methods are important given their rapid initial progress, relative insensitivity to hyperparameters, and ability to deal with ill-scaled problems. Several recent papers have attempted to explain and improve adaptive methods (Loshchilov & Hutter, 2017; Anonymous, 2018; Zhang et al., 2017). However, given that they retain the adaptivity and non-uniform gradient scaling, they too

[2] pytorch.org



(a) ResNet-32 — CIFAR-10 (b) DenseNet — CIFAR-10 (c) PyramidNet — CIFAR-10 (d) SENet — CIFAR-10



(e) ResNet-32 — CIFAR-100 (f) DenseNet — CIFAR-100 (g) PyramidNet — CIFAR-100 (h) SENet — CIFAR-100

Figure 4. Numerical experiments comparing SGD(M), Adam and SWATS with tuned learning rates on the ResNet-32, DenseNet, PyramidNet and SENet architectures on CIFAR-10 and CIFAR-100 data sets.

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†	0.003	0.003	7.52	186.03
QRNN	PTB	35†	0.002	0.002	4.61	184.14
LSTM	WT-2	60†	0.003	0.003	1.11	259.47
QRNN	WT-2	60†	0.003	0.004	14.46	295.71
ResNet-18	Tiny-ImageNet	0.2	0.001	0.0007	1.71	48.91

Table 1. Summarizing the optimal hyperparameters for SGD(M), Adam and SWATS for all experiments and, in the case of SWATS, the value of the estimated learning rate for SGD after the switch and the switchover point in epochs. † denotes that no momentum was employed for SGDM.

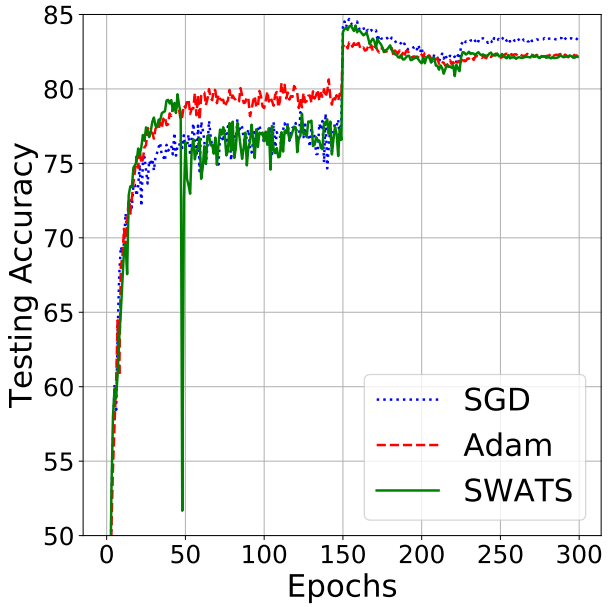


Figure 5. Numerical experiments comparing SGD(M), Adam and SWATS with tuned learning rates on the ResNet-18 architecture on the Tiny-ImageNet data set.

are expected to suffer from similar generalization issues as Adam. Motivated by this observation, we investigate the question of using a hybrid training strategy that starts with an adaptive method and switches to SGD. By design, both the switchover point and the learning rate for SGD after the switch, are determined as a part of the algorithm and as such require no added tuning effort. We demonstrate the efficacy of this approach on several standard benchmarks, including a host of architectures, on the PennTree Bank, WikiText-2, Tiny-ImageNet, CIFAR-10 and CIFAR-100 data sets. In summary, our results show that the proposed strategy leads to results comparable to SGD while retaining the beneficial properties of Adam such as hyperparameter insensitivity and rapid initial progress.

The success of our strategy motivates a deeper exploration into the interplay between the dynamics of the optimizer and the generalization performance. Recent theoretical work analyzing generalization for deep learning suggests coupling generalization arguments with the training process (Soudry et al., 2017; Hardt et al., 2015; Zhang et al., 2016; Wilson et al., 2017). The optimizers choose different trajectories in the parameter space and are attracted to different basins of attractions, with vastly different generalization performance. Even for a simple least-squares problem: $\min_w \|Xw - y\|_2^2$ with $w_0 = 0$, SGD recovers the minimum-norm solution, with its associated margin benefits, whereas adaptive methods do not. The fundamental reason for this is that SGD ensures that the iterates remain in the column space of the X, and that only one optimum exists in that column space, viz. the minimum-norm solution. On the other hand, adaptive methods do not necessarily stay in the column space of X. Similar arguments can be constructed for logistic regression problems (Soudry et al., 2017), but an analogous treatment for deep networks is, to the best of our knowledge, an open question. We hypothesize that a successful implementation of a hybrid strategy, such as SWATS, suggests that in the case of deep networks, despite training for few epochs before switching to SGD, the model is able to navigate towards a basin with better generalization performance. However, further empirical and theoretical evidence is necessary to buttress this hypothesis, and is a topic of future research.

While the focus of this work has been on Adam, the strategy proposed is generally applicable and can be analogously employed to other adaptive methods such as Adagrad and RMSProp. A viable research direction includes exploring the possibility of switching back-and-forth, as needed, from Adam to SGD. Indeed, in our preliminary experiments, we found that switching back from SGD to Adam at the end of a 300 epoch run for any of the experiments on the CIFAR-10 data set yielded slightly better performance. Along the same line, a future research direction includes a smoother transition from Adam to SGD as opposed to the hard switch proposed in this paper, which may cause short-term performance degradation. This can be achieved by using a convex combination of the SGD and Adam directions as in the case of Akiba et al. (2017), and gradually increasing the weight for the SGD contribution by a criterion. Finally, we note that the strategy proposed in this paper does not preclude the use of those proposed in Zhang et al. (2017); Loshchilov & Hutter (2017); Anonymous (2018). We plan to investigate the performance of the algorithm obtained by mixing these strategies, such as monotonic increase guarantees of the second-order moment, cosine-annealing, ℓ_2-norm correction, in the future.

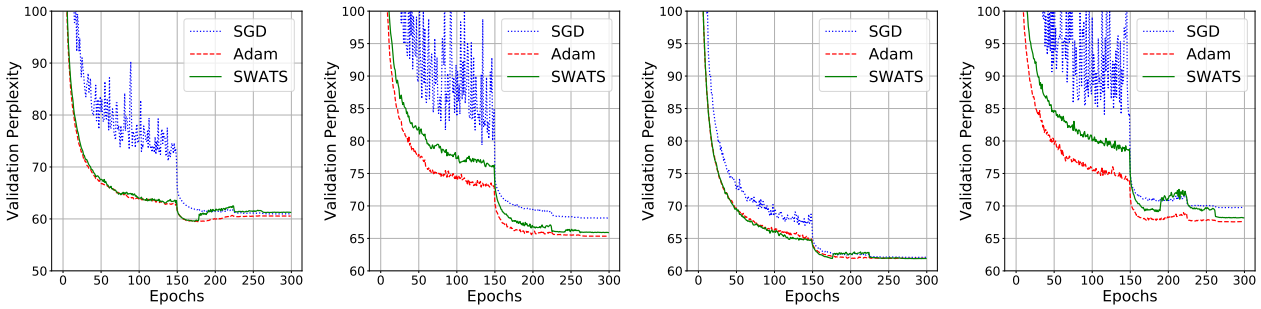
References

Akiba, T., Suzuki, S., and Fukuda, K. Extremely large minibatch SGD: Training resnet-50 on ImageNet in 15 minutes. *arXiv preprint arXiv:1711.04325*, 2017.

Anonymous. On the convergence of Adam and beyond. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

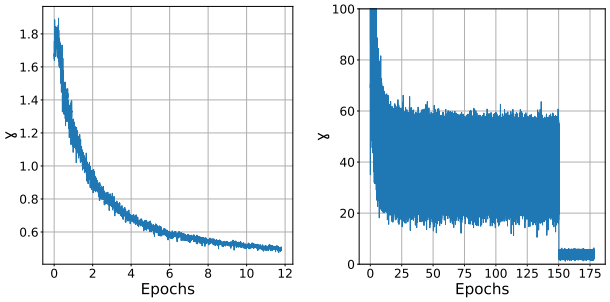
Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bradbury, J., Merity, S., Xiong, C., and Socher, R.



(a) LSTM — PTB (b) LSTM — WT2 (c) QRNN — PTB (d) QRNN — WT2

Figure 6. Numerical experiments comparing SGD(M), Adam and SWATS with tuned learning rates on the AWD-LSTM and AWD-QRNN architectures on PTB and WT-2 data sets.



(a) DenseNet — CIFAR-100 (b) QRNN — PTB

Figure 7. Evolution of the estimated SGD learning rate (γ_k) on two representative tasks.

Quasi-Recurrent Neural Networks. *arXiv preprint arXiv:1611.01576*, 2016.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Han, D., Kim, J., and Kim, J. Deep pyramidal residual networks. *arXiv preprint arXiv:1610.02915*, 2016.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., and Keutzer, K. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

Karpathy, A. A Peek at Trends in Machine Learning. https://medium.com/@karpathy/a-peek-at-trends-in-machine-learning-ab8a1085a106, 2017. [Online; accessed 12-Dec-2017].

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR 2015)*, 2015.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Lee, J., Simchowitz, M., Jordan, M. I, and Recht, B. Gradient descent converges to minimizers. *University of California, Berkeley*, 1050:16, 2016.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. 2016.

Loshchilov, I. and Hutter, F. Fixing Weight Decay Regularization in Adam. *ArXiv e-prints*, November 2017.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic Gradient Descent as Approximate Bayesian Inference. *ArXiv e-prints*, April 2017.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Merity, S., Keskar, N., and Socher, R. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*, 2017.

Mikolov, T., Kombrink, S., Deoras, A., Burget, L., and Cernocky, J. RNNLM-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pp. 196–201, 2011.

Nocedal, J. and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

Robbins, Herbert and Monro, Sutton. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Soudry, D., Hoffer, E., and Srebro, N. The implicit bias of gradient descent on separable data. *arXiv preprint arXiv:1710.10345*, 2017.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Tieleman, T. and Hinton, G. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y, and Fergus, R. Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pp. 1058–1066, 2013.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The Marginal Value of Adaptive Gradient Methods in Machine Learning. *ArXiv e-prints*, May 2017.

Wu, Y., Schuster, M., Chen, Z., Le, Q., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, Z., Ma, L., Li, Z., and Wu, C. Normalized direction-preserving Adam. *arXiv preprint arXiv:1709.04546*, 2017.