Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks

Jinghui Chen[1] , **Dongruo Zhou**[1] , **Yiqi Tang**[2] , **Ziyan Yang**[3] , **Yuan Cao**[1] and **Quanquan Gu**[1]
[1]University of California, Los Angeles
[2]Ohio State University
[3]University of Virginia
{jhchen, drzhou, yuancao, qgu}@cs.ucla.edu, tang.1466@osu.edu, zy3cx@virginia.edu

Abstract

Adaptive gradient methods, which adopt historical gradient information to automatically adjust the learning rate, despite the nice property of fast convergence, have been observed to generalize worse than stochastic gradient descent (SGD) with momentum in training deep neural networks. This leaves how to close the generalization gap of adaptive gradient methods an open problem. In this work, we show that adaptive gradient methods such as Adam, Amsgrad, are sometimes "over adapted". We design a new algorithm, called *Partially adaptive momentum estimation method*, which unifies the Adam/Amsgrad with SGD by introducing a partial adaptive parameter p, to achieve the best from both worlds. We also prove the convergence rate of our proposed algorithm to a stationary point in the stochastic nonconvex optimization setting. Experiments on standard benchmarks show that our proposed algorithm can maintain fast convergence rate as Adam/Amsgrad while generalizing as well as SGD in training deep neural networks. These results would suggest practitioners pick up adaptive gradient methods once again for faster training of deep neural networks.

1 Introduction

Stochastic gradient descent (SGD) is now one of the most dominant approaches for training deep neural networks [Goodfellow *et al.*, 2016]. In each iteration, SGD only performs one parameter update on a mini-batch of training examples. SGD is simple and has been proved to be efficient, especially for tasks on large datasets. In recent years, adaptive variants of SGD have emerged and shown their successes for their convenient automatic learning rate adjustment mechanism. Adagrad [Duchi *et al.*, 2011] is probably the first along this line of research, and significantly outperforms vanilla SGD in the sparse gradient scenario. Despite the first success, Adagrad was later found to demonstrate degraded performance especially in cases where the loss function is nonconvex or the gradient is dense. Many variants of Adagrad, such as RMSprop [Hinton *et al.*, 2012], Adam [Kingma and Ba, 2015], Adadelta [Zeiler, 2012], Nadam [Dozat, 2016], were then proposed to address these challenges by adopting exponential moving average rather than the arithmetic average used in Adagrad. This change largely mitigates the rapid decay of learning rate in Adagrad and hence makes this family of algorithms, especially Adam, particularly popular on various tasks. Recently, it has also been observed [Reddi *et al.*, 2018] that Adam does not converge in some settings where rarely encountered large gradient information quickly dies out due to the "short momery" problem of exponential moving average. To address this issue, Amsgrad [Reddi *et al.*, 2018] has been proposed to keep an extra "long term memory" variable to preserve the past gradient information and to correct the potential convergence issue in Adam. There are also some other variants of adaptive gradient method such as SC-Adagrad / SC-RMSprop [Mukkamala and Hein, 2017], which derives logarithmic regret bounds for strongly convex functions.

On the other hand, people recently found that for largely over-parameterized neural networks, e.g., more complex modern convolutional neural network (CNN) architectures such as VGGNet [He *et al.*, 2016], ResNet [He *et al.*, 2016], Wide ResNet [Zagoruyko and Komodakis, 2016], DenseNet [Huang *et al.*, 2017], training with Adam or its variants typically generalizes worse than SGD with momentum, even when the training performance is better. In particular, people found that carefully-tuned SGD, combined with proper momentum, weight decay and appropriate learning rate decay schedules, can significantly outperform adaptive gradient algorithms eventually [Wilson *et al.*, 2017]. As a result, many recent studies train their models with SGD-Momentum [He *et al.*, 2016; Zagoruyko and Komodakis, 2016; Huang *et al.*, 2017; Simonyan and Zisserman, 2014; Ren *et al.*, 2015; Xie *et al.*, 2017; Howard *et al.*, 2017] despite that adaptive gradient algorithms usually converge faster. Different from SGD, which adopts a universal learning rate for all coordinates, the effective learning rate of adaptive gradient methods, i.e., the universal base learning rate divided by the second order moment term, is different for different coordinates. Due to the normalization of the second order moment, some coordinates will have very large effective learning rates. To alleviate this problem, one usually chooses a smaller base learning rate for adaptive gradient methods than SGD with momentum. This makes the learning rate decay schedule less effective when applied to adaptive gradient methods, since a much

smaller base learning rate will lead to diminishing effective learning rate for most coordinates after several rounds of decay. We refer to the above phenomenon as the "small learning rate dilemma" (see more details in Section 3).

With all these observations, a natural question is:

Can we take the best from both Adam and SGD-Momentum, i.e., design an algorithm that not only enjoys the fast convergence rate as Adam, but also generalizes as well as SGD-Momentum?

In this paper, we answer this question affirmatively. We close the generalization gap of adaptive gradient methods by proposing a new algorithm, called **p**artially **ada**ptive **m**omentum estimation (Padam) method, which unifies Adam/Amsgrad with SGD-Momentum to achieve the best of both worlds, by a partially adaptive parameter. The intuition behind our algorithm is: by controlling the degree of adaptiveness, the base learning rate in Padam does not need to be as small as other adaptive gradient methods. Therefore, it can maintain a larger learning rate while preventing the gradient explosion. We note that there exist several studies [Zaheer *et al.*, 2018; Loshchilov and Hutter, 2019; Luo *et al.*, 2019] that also attempted to address the same research question. In detail, Yogi [Zaheer *et al.*, 2018] studied the effect of adaptive denominator constant ϵ and mini-batch size in the convergence of adaptive gradient methods. AdamW [Loshchilov and Hutter, 2019] proposed to fix the weight decay regularization in Adam by decoupling the weight decay from the gradient update and this improves the generalization performance of Adam. AdaBound [Luo *et al.*, 2019] applies dynamic bound of learning rate on Adam and make them smoothly converge to a constant final step size as in SGD. Our algorithm is very different from Yogi, AdamW and AdaBound. Padam is built upon a simple modification of Adam without extra complicated algorithmic design and it comes with a rigorous convergence guarantee in the nonconvex stochastic optimization setting.

We highlight the main contributions of our work as follows:

• We propose a novel and simple algorithm Padam with a partially adaptive parameter, which resolves the "small learning rate dilemma" for adaptive gradient methods and allows for faster convergence, hence closing the gap of generalization.

• We provide a convergence guarantee for Padam in nonconvex optimization. Specifically, we prove that the convergence rate of Padam to a stationary point for stochastic nonconvex optimization is

$$O\bigg(\frac{d^{1/2}}{T^{3/4-s/2}}+\frac{d}{T}\bigg), \tag{1.1}$$

where s characterizes the growth rate of the cumulative stochastic gradient $\mathbf{g}_{1:T,i} = [g_{1,i}, g_{2,i}, \ldots, g_{T,i}]^\top$ ($\mathbf{g}_1, \ldots, \mathbf{g}_T$ are the stochastic gradients) and $0 \le s \le 1/2$. When the stochastic gradients are sparse, i.e., $s < 1/2$, (1.1) is strictly better than the convergence rate of SGD in terms of the rate of T.

• We also provide thorough experiments about our proposed Padam method on training modern deep neural architectures. We empirically show that Padam achieves the fastest convergence speed while generalizing as well as SGD with momentum. These results suggest that practitioners should pick up adaptive gradient methods once again for faster training of deep neural networks.

• Last but not least, compared with the recent work on adaptive gradient methods, such as Yogi [Zaheer *et al.*, 2018], AdamW [Loshchilov and Hutter, 2019], AdaBound [Luo *et al.*, 2019], our proposed Padam achieves better generalization performance than these methods in our experiments.

1.1 Additional Related Work

Here we review additional related work that is not covered before. [Zhang *et al.*, 2017] proposed a normalized direction-preserving Adam (ND-Adam), which changes the adaptive terms from individual dimensions to the whole gradient vector. [Keskar and Socher, 2017] proposed to improve the generalization performance by switching from Adam to SGD. On the other hand, despite the great successes of adaptive gradient methods for training deep neural networks, the convergence guarantees for these algorithms are still understudied. Most convergence analyses of adaptive gradient methods are restricted to online convex optimization [Duchi *et al.*, 2011; Kingma and Ba, 2015; Mukkamala and Hein, 2017; Reddi *et al.*, 2018]. A few recent attempts have been made to analyze adaptive gradient methods for stochastic nonconvex optimization. More specifically, [Basu *et al.*, 2018] proved the convergence rate of RMSProp and Adam when using deterministic gradient rather than stochastic gradient. [Li and Orabona, 2018] analyzed convergence rate of AdaGrad under both convex and nonconvex settings but did not consider more complicated Adam-type algorithms. [Ward *et al.*, 2018] also proved the convergence rate of AdaGrad under both convex and nonconvex settings without considering the effect of stochastic momentum. [Chen *et al.*, 2018] provided a convergence analysis for a class of Adam-type algorithms for nonconvex optimization. [Zou and Shen, 2018] analyzed the convergence rate of AdaHB and AdaNAG, two modified version of AdaGrad with the use of momentum. [Liu *et al.*, 2019] proposed Optimistic Adagrad and showed its convergence in non-convex non-concave min-max optimization. However, none of these results are directly applicable to Padam. Our convergence analysis in Section 4 is quite general and implies the convergence rate of AMSGrad for nonconvex optimization. In terms of learning rate decay schedule, [Wu *et al.*, 2018] studied the learning rate schedule via short-horizon bias. [Xu *et al.*, 2016; Davis *et al.*, 2019] analyzed the convergence of stochastic algorithms with geometric learning rate decay. [Ge *et al.*, 2019] studied the learning rate schedule for quadratic functions.

The remainder of this paper is organized as follows: in Section 2, we briefly review existing adaptive gradient methods. We present our proposed algorithm in Section 3, and the main theory in Section 4. In Section 5, we compare the proposed algorithm with existing algorithms on modern neural network architectures on benchmark datasets. Finally, we conclude this paper and point out the future work in Section 6.

Notation: Scalars are denoted by lower case letters, vectors by lower case bold face letters, and matrices by upper case bold face letters. For a vector $\mathbf{x} \in \mathbb{R}^d$, we denote

the ℓ_2 norm of $\mathbf{x}$ by $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d x_i^2}$, the ℓ_∞ norm of $\mathbf{x}$ by $\|\mathbf{x}\|_\infty = \max_{i=1}^d |x_i|$. For a sequence of vectors $\{\mathbf{x}_j\}_{j=1}^t$, we denote by $x_{j,i}$ the i-th element in $\mathbf{x}_j$. We also denote $\mathbf{x}_{1:t,i} = [x_{1,i}, \ldots, x_{t,i}]^\top$. With slight abuse of notation, for two vectors $\mathbf{a}$ and $\mathbf{b}$, we denote $\mathbf{a}^2$ as the element-wise square, $\mathbf{a}^p$ as the element-wise power operation, $\mathbf{a}/\mathbf{b}$ as the element-wise division and $\max(\mathbf{a}, \mathbf{b})$ as the element-wise maximum. We denote by $\mathrm{diag}(\mathbf{a})$ a diagonal matrix with diagonal entries $a_1, \ldots, a_d$. Given two sequences $\{a_n\}$ and $\{b_n\}$, we write $a_n = O(b_n)$ if there exists a positive constant C such that $a_n \leq Cb_n$ and $a_n = o(b_n)$ if $a_n/b_n \to 0$ as $n \to \infty$. Notation $\widetilde{O}(\cdot)$ hides logarithmic factors.

2 Review of Adaptive Gradient Methods

Various adaptive gradient methods have been proposed in order to achieve better performance on various stochastic optimization tasks. Adagrad [Duchi *et al.*, 2011] is among the first methods with adaptive learning rate for each individual dimension, which motivates the research on adaptive gradient methods in the machine learning community. In detail, Adagrad[1] adopts the following update form:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t \frac{\mathbf{g}_t}{\sqrt{\mathbf{v}_t}}, \text{ where } \mathbf{v}_t = \frac{1}{t}\sum_{j=1}^t \mathbf{g}_j^2,$$

where $\mathbf{g}_t$ stands for the stochastic gradient $\nabla f_t(\mathbf{x}_t)$, and $\alpha_t = \alpha/\sqrt{t}$ is the step size. In this paper, we call α_t *base learning rate*, which is the same for all coordinates of $\mathbf{x}_t$, and we call $\alpha_t/\sqrt{v_{t,i}}$ *effective learning rate* for the i-th coordinate of $\mathbf{x}_t$, which varies across the coordinates. Adagrad is proved to enjoy a huge gain in terms of convergence especially in sparse gradient situations. Empirical studies also show a performance gain even for non-sparse gradient settings. RMSprop [Hinton *et al.*, 2012] follows the idea of adaptive learning rate and it changes the arithmetic averages used for $\mathbf{v}_t$ in Adagrad to exponential moving averages. Even though RMSprop is an empirical method with no theoretical guarantee, the outstanding empirical performance of RMSprop raised people's interests in exponential moving average variants of Adagrad. Adam [Kingma and Ba, 2015][2] is the most popular exponential moving average variant of Adagrad. It combines the idea of RMSprop and momentum acceleration, and takes the following update form:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t \frac{\mathbf{m}_t}{\sqrt{\mathbf{v}_t}} \text{ where}$$
$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t, \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2.$$

Adam also requires $\alpha_t = \alpha/\sqrt{t}$ for the sake of convergence analysis. In practice, any decaying step size or even constant step size works well for Adam. Note that if we choose $\beta_1 = 0$, Adam basically reduces to RMSprop. [Reddi *et al.*, 2018] identified a non-convergence issue in Adam. Specifically, Adam does not collect long-term memory of past gradients and therefore the effective learning rate could be increasing in some cases. They proposed a modified algorithm namely Amsgrad. More specifically, Amsgrad adopts an additional step to ensure the decay of the effective learning rate $\alpha_t/\sqrt{\widehat{\mathbf{v}}_t}$, and its key update formula is as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t \frac{\mathbf{m}_t}{\sqrt{\widehat{\mathbf{v}}_t}}, \text{ where } \widehat{\mathbf{v}}_t = \max(\widehat{\mathbf{v}}_{t-1}, \mathbf{v}_t),$$

$\mathbf{m}_t$ and $\mathbf{v}_t$ are the same as Adam. By introducing the $\widehat{\mathbf{v}}_t$ term, [Reddi *et al.*, 2018] corrected some mistakes in the original proof of Adam and proved an $O(1/\sqrt{T})$ convergence rate of Amsgrad for convex optimization. Note that all the theoretical guarantees on adaptive gradient methods (Adagrad, Adam, Amsgrad) are only proved for convex functions.

Algorithm 1 Partially adaptive momentum estimation method (Padam)

input: initial point $\mathbf{x}_1 \in \mathcal{X}$; step sizes $\{\alpha_t\}$; adaptive parameters $\beta_1, \beta_2, p \in (0, 1/2]$
set $\mathbf{m}_0 = \mathbf{0}$, $\mathbf{v}_0 = \mathbf{0}$, $\widehat{\mathbf{v}}_0 = \mathbf{0}$
for $t = 1, \ldots, T$ **do**
  $\mathbf{g}_t = \nabla f(\mathbf{x}_t, \xi_t)$
  $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$
  $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2$
  $\widehat{\mathbf{v}}_t = \max(\widehat{\mathbf{v}}_{t-1}, \mathbf{v}_t)$
  $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t \cdot \mathbf{m}_t / \widehat{\mathbf{v}}_t^p$
end for
Output: Choose $\mathbf{x}_{\text{out}}$ from $\{\mathbf{x}_t\}, 2 \leq t \leq T$ with probability $\alpha_{t-1}/\big(\sum_{i=1}^{T-1} \alpha_i\big)$

3 The Proposed Algorithm

In this section, we propose a new algorithm for bridging the generalization gap for adaptive gradient methods. Specifically, we introduce a partial adaptive parameter p to control the level of adaptiveness of the optimization procedure. The proposed algorithm is displayed in Algorithm 1.

In Algorithm 1, $\mathbf{g}_t$ denotes the stochastic gradient and $\widehat{\mathbf{v}}_t$ can be seen as a moving average over the second order moment of the stochastic gradients. As we can see from Algorithm 1, the key difference between Padam and Amsgrad [Reddi *et al.*, 2018] is that: while $\mathbf{m}_t$ is still the momentum as in Adam/Amsgrad, it is now "partially adapted" by the second order moment. We call $p \in [0, 1/2]$ the partially adaptive parameter. Note that $1/2$ is the largest possible value for p and a larger p will result in non-convergence in the proof (see the proof details in the supplementary materials). When $p \to 0$, Algorithm 1 reduces to SGD with momentum[3] and when $p = 1/2$, Algorithm 1 is exactly Amsgrad. Therefore, Padam indeed unifies Amsgrad and SGD with momentum.

With the notations defined above, we are able to formally explain the "small learning rate dilemma". In order to make

[1] The formula is equivalent to the one from the original paper [Duchi *et al.*, 2011] after simple manipulations.

[2] Here for simplicity and consistency, we ignore the bias correction step in the original paper of Adam. Yet adding the bias correction step will not affect the argument in the paper.

[3] The only difference between Padam with $p = 0$ and SGD-Momentum is an extra constant factor $(1-\beta_1)$, which can be moved into the learning rate such that the update rules for these two algorithms are identical.

things clear, we first emphasize the relationship between adaptiveness and learning rate decay. We refer the actual learning rate applied to $\mathbf{m}_t$ as the effective learning rate, i.e., $\alpha_t/\widehat{\mathbf{v}}_t^p$. Now suppose that a learning rate decay schedule is applied to α_t. If p is large, then at early stages, the effective learning rate $\alpha_t/\widehat{v}_{t,i}^p$ could be fairly large for certain coordinates with small $\widehat{v}_{t,i}$ value[4]. To prevent those coordinates from overshooting we need to enforce a smaller α_t, and therefore the base learning rate must be set small [Keskar and Socher, 2017; Wilson *et al.*, 2017]. As a result, after several rounds of decaying, the learning rates of the adaptive gradient methods are too small to make any significant progress in the training process[5]. We call this phenomenon "small learning rate dilemma". It is also easy to see that the larger p is, the more severe "small learning rate dilemma" is. This suggests that intuitively, we should consider using Padam with a proper adaptive parameter p, and choosing $p < 1/2$ can potentially make Padam suffer less from the "small learning rate dilemma" than Amsgrad, which justifies the range of p in Algorithm 1. We will show in our experiments (Section 5) that Padam with $p < 1/2$ can adopt an equally large base learning rate as SGD with momentum.

Note that even though in Algorithm 1, the choice of α_t covers different choices of learning rate decay schedule, the main focus of this paper is not about finding the best learning rate decay schedule, but designing a new algorithm to control the adaptiveness for better empirical generalization result. In other words, our focus is not on α_t, but on $\widehat{\mathbf{v}}_t$. For this reason, we simply fix the learning rate decay schedule for all methods in the experiments to provide a fair comparison for different methods.

Figure 1 shows the comparison of test error performances under the different partial adaptive parameter p for ResNet on both CIFAR-10 and CIFAR-100 datasets. We can observe that a larger p will lead to fast convergence at early stages and worse generalization performance later, while a smaller p behaves more like SGD with momentum: slow in early stages but finally catch up. With a proper choice of p (e.g., $1/8$ in this case), Padam can obtain the best of both worlds. Note that besides Algorithm 1, our partially adaptive idea can also be applied to other adaptive gradient methods such as Adagrad, Adadelta, RMSprop, AdaMax [Kingma and Ba, 2015]. For the sake of conciseness, we do not list the partially adaptive versions for other adaptive gradient methods here. We also would like to comment that Padam is totally different from the p-norm generalized version of Adam in [Kingma and Ba, 2015], which induces AdaMax method when $p \to \infty$. In their case, p-norm is used to generalize 2-norm of their current and past gradients while keeping the scale of adaptation unchanged. In sharp contrast, we intentionally change (reduce) the scale of the adaptive term in Padam to get better generalization performance.

[4]The coordinate $\widehat{v}_{t,i}$'s are much less than 1 for most commonly used network architectures.

[5]This does not mean the learning rate decay schedule weakens adaptive gradient method. On the contrary, applying the learning rate decay schedule still gives performance boost to the adaptive gradient methods in general but this performance boost is not as significant as SGD + momentum.

4 Convergence Analysis of the Proposed Algorithm

In this section, we establish the convergence theory of Algorithm 1 in the stochastic nonconvex optimization setting, i.e., we aim at solving the following stochastic nonconvex optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) := \mathbb{E}_\xi\big[f(\mathbf{x};\xi)\big],$$

where ξ is a random variable satisfying certain distribution, $f(\mathbf{x};\xi):\mathbb{R}^d\to\mathbb{R}$ is a L-smooth nonconvex function. In the stochastic setting, one cannot directly access the full gradient of $f(\mathbf{x})$. Instead, one can only get unbiased estimators of the gradient of $f(\mathbf{x})$, which is $\nabla f(\mathbf{x};\xi)$. This setting has been studied in [Ghadimi and Lan, 2013; Ghadimi and Lan, 2016]. We first introduce the following assumptions.

Assumption 4.1 (Bounded Gradient)**.** $f(\mathbf{x}) = \mathbb{E}_\xi f(\mathbf{x};\xi)$ has G_∞-bounded stochastic gradient. That is, for any ξ, we assume that $\|\nabla f(\mathbf{x};\xi)\|_\infty \le G_\infty$.

It is worth mentioning that Assumption 4.1 is slightly weaker than the ℓ_2-boundedness assumption $\|\nabla f(\mathbf{x};\xi)\|_2 \le G_2$ used in [Reddi *et al.*, 2016; Chen *et al.*, 2018]. Since $\|\nabla f(\mathbf{x};\xi)\|_\infty \le \|\nabla f(\mathbf{x};\xi)\|_2 \le \sqrt{d}\|\nabla f(\mathbf{x};\xi)\|_\infty$, the ℓ_2-boundedness assumption implies Assumption 4.1 with $G_\infty = G_2$. Meanwhile, G_∞ will be tighter than G_2 by a factor of $\sqrt{d}$ when each coordinate of $\nabla f(\mathbf{x};\xi)$ almost equals to each other.

Assumption 4.2 (L-smooth)**.** $f(\mathbf{x}) = \mathbb{E}_\xi f(\mathbf{x};\xi)$ is L-smooth: for any $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$, it satisfied that $\big|f(\mathbf{x}) - f(\mathbf{y}) - \langle\nabla f(\mathbf{y}),\mathbf{x}-\mathbf{y}\rangle\big| \le \frac{L}{2}\|\mathbf{x}-\mathbf{y}\|_2^2$.

Assumption 4.2 is frequently used in analysis of gradient-based algorithms. It is equivalent to the L-gradient Lipschitz condition, which is often written as $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \le L\|\mathbf{x}-\mathbf{y}\|_2$. Next we provide the main convergence rate result for our proposed algorithm. The detailed proof can be found in the longer version of this paper.

Theorem 4.3. In Algorithm 1, suppose that $p \in [0, 1/2]$, $\beta_1 < \beta_2^{2p}$, $\alpha_t = \alpha$ and $\|\mathbf{g}_{1:T,i}\|_2 \le G_\infty T^s$ for $t = 1,\ldots,T$, $0 \le s \le 1/2$, under Assumptions 4.1 and 4.2, let $\Delta f = f(\mathbf{x}_1) - \inf_{\mathbf{x}} f(\mathbf{x})$, for any $q \in [\max\{0, 4p-1\}, 1]$, the output $\mathbf{x}_{\text{out}}$ of Algorithm 1 satisfies that

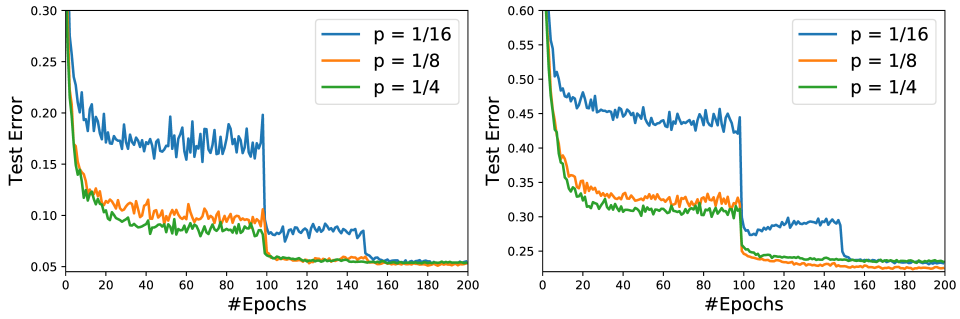
$$\mathbb{E}\Big[\big\|\nabla f(\mathbf{x}_{\text{out}})\big\|_2^2\Big] \le \frac{M_1}{T\alpha} + \frac{M_2 d}{T} + \frac{M_3\alpha d G_\infty^{1-q}}{T^{(1-q)(1/2-s)}}, \tag{4.1}$$

where

$$M_1 = 2G_\infty^{2p}\Delta f,\ M_2 = \frac{4G_\infty^{2+2p}\mathbb{E}\big\|\widehat{\mathbf{v}}_1^{-p}\big\|_1}{d(1-\beta_1)} + 4G_\infty^2,$$

$$M_3 = \frac{4LG_\infty^{1+q-2p}}{(1-\beta_2)^{2p}} + \frac{8LG_\infty^{1+q-2p}(1-\beta_1)}{(1-\beta_2)^{2p}(1-\beta_1/\beta_2^{2p})}\bigg(\frac{\beta_1}{1-\beta_1}\bigg)^2.$$

Remark 4.4. From Theorem 4.3, we can see that M_1 and M_3 are independent of the number of iterations T and dimension d. In addition, if $\|\widehat{\mathbf{v}}_1^{-1}\|_\infty = O(1)$, it is easy to see that M_2 also has an upper bound that is independent of T and d. s



(a) CIFAR-10 (b) CIFAR-100

Figure 1: Performance comparison of Padam with different choices of p for training ResNet on (a) CIFAR-10 and (b) CIFAR-100 datasets.

characterizes the growth rate condition [Liu *et al.*, 2019] of the cumulative stochastic gradient $\mathbf{g}_{1:T,i}$. In the worse case, $s = 1/2$, while in practice when the stochastic gradients are sparse, $s < 1/2$.

The following corollary simplifies the result of Theorem 4.3 by choosing $q = 0$ under the condition $p \in [0, 1/4]$.

Corollary 4.5. Under the same conditions in Theorem 4.3, if $p \in [0, 1/4]$, Padam's output satisfies

$$\mathbb{E}\Big[\big\|\nabla f(\mathbf{x}_{\text{out}})\big\|_2^2\Big] \leq \frac{M_1}{T\alpha} + \frac{M_2 d}{T} + \frac{M_3' \alpha d G_\infty}{T^{1/2-s}}, \tag{4.2}$$

where M_1 and M_2 and Δf are the same as in Theorem 4.3, and M_3' is defined as follows:

$$M_3' = \frac{4LG_\infty^{1-2p}}{(1-\beta_2)^{2p}} + \frac{8LG_\infty^{1-2p}(1-\beta_1)}{(1-\beta_2)^{2p}(1-\beta_1/\beta_2^{2p})}\left(\frac{\beta_1}{1-\beta_1}\right)^2.$$

Remark 4.6. We show the convergence rate under optimal choice of step size α. If

$$\alpha = \Theta\big(d^{1/2}T^{1/4+s/2}\big)^{-1},$$

then by (4.2), we have

$$\mathbb{E}\Big[\big\|\nabla f(\mathbf{x}_{\text{out}})\big\|_2^2\Big] = O\bigg(\frac{d^{1/2}}{T^{3/4-s/2}} + \frac{d}{T}\bigg). \tag{4.3}$$

Note that the convergence rate given by (4.3) is related to s. In the worst case when $s = 1/2$, we have $\mathbb{E}[\|\nabla f(\mathbf{x}_{\text{out}})\|_2^2] = O\big(\sqrt{d/T} + d/T\big)$, which matches the rate $O(1/\sqrt{T})$ achieved by nonconvex SGD [Ghadimi and Lan, 2016], considering the dependence of T. When the stochastic gradients $\mathbf{g}_{1:T,i}$, $i = 1, \ldots, d$ are sparse, i.e., $s < 1/2$, the convergence rate in (4.3) is strictly better than the convergence rate of nonconvex SGD [Ghadimi and Lan, 2016].

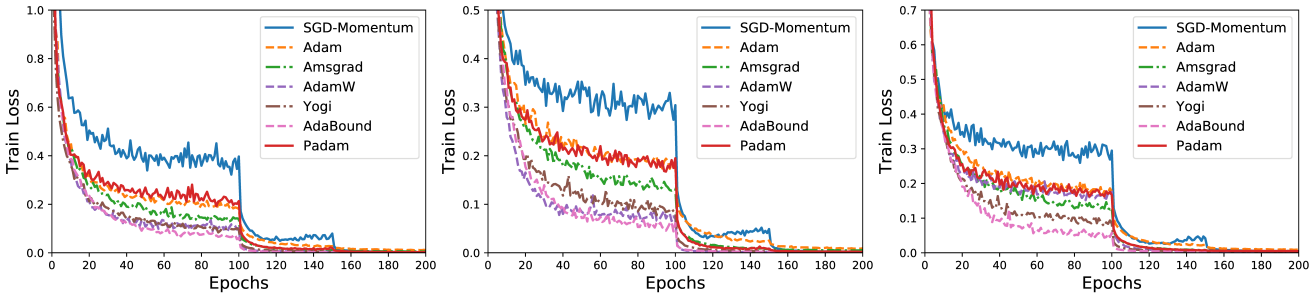
5 Experiments

In this section, we empirically evaluate our proposed algorithm for training various modern deep learning models and test them on several standard benchmarks.[6] We show that for nonconvex loss functions in deep learning, our proposed algorithm still enjoys a fast convergence rate, while its generalization performance is as good as SGD with momentum and much better than existing adaptive gradient methods such as Adam and Amsgrad.

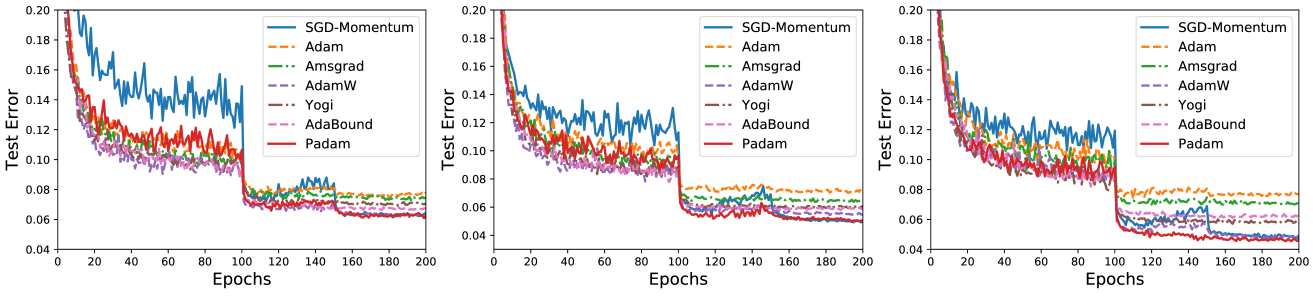
We compare Padam against several state-of-the-art algorithms, including: (1) SGD-Momentum, (2) Adam [Kingma and Ba, 2015], (3) Amsgrad [Reddi *et al.*, 2018], (4) AdamW [Loshchilov and Hutter, 2019] (5) Yogi [Zaheer *et al.*, 2018] and (6) AdaBound [Luo *et al.*, 2019]. We use several popular datasets for image classifications and language modeling: CIFAR-10 [Krizhevsky and Hinton, 2009], ImageNet dataset (ILSVRC2012) [Deng *et al.*, 2009] and Penn Treebank dataset [Marcus *et al.*, 1993]. We adopt three popular CNN architectures for image classification task: VGGNet-16 [Simonyan and Zisserman, 2014], Residual Neural Network (ResNet-18) [He *et al.*, 2016], Wide Residual Network (WRN-16-4) [Zagoruyko and Komodakis, 2016]. We test the language modeling task using 2-layer and 3-layer Long Short-Term Memory (LSTM) network [Hochreiter and Schmidhuber, 1997]. For CIFAR-10 and Penn Treebank experiments, we test for 200 epochs and decay the learning rate by 0.1 at the 100th and 150th epoch. We test ImageNet tasks for 100 epochs with similar multi-stage learning rate decaying scheme at the 30th, 60th and 80th epoch.

We perform grid searches to choose the best hyperparameters for all algorithms in both image classification and language modeling tasks. For the base learning rate, we do grid search over $\{10^{-4}, \ldots, 10^2\}$ for all algorithms, and choose the partial adaptive parameter p from $\{2/5, 1/4, 1/5, 1/8, 1/16\}$ and the second order moment parameter β_2 from $\{0.9, 0.99, 0.999\}$. For image classification experiments, we set the base learning rate of 0.1 for SGD with momentum and Padam, 0.001 for all other adaptive gradient methods. β_1 is set as 0.9 for all methods. β_2 is set as 0.99 for Adam and Amsgrad, 0.999 for all other methods. For Padam, the partially adaptive parameter p is set to be $1/8$. For AdaBound, the final learning rate is set to be 0.1. For AdamW, the normalized weight decay factor is set to 2.5×10^{-2} for CIFAR-10 and 5×10^{-2} for ImageNet. For Yogi, ϵ is set as 10^{-3} as suggested in the original paper. The minibatch size for CIFAR-10 is set to be 128 and for ImageNet dataset we set it to be 256. Regarding the LSTM experiments, for SGD with momentum, the base learning rate is set to be 1 for

[6] The code is available at https://github.com/uclaml/Padam.

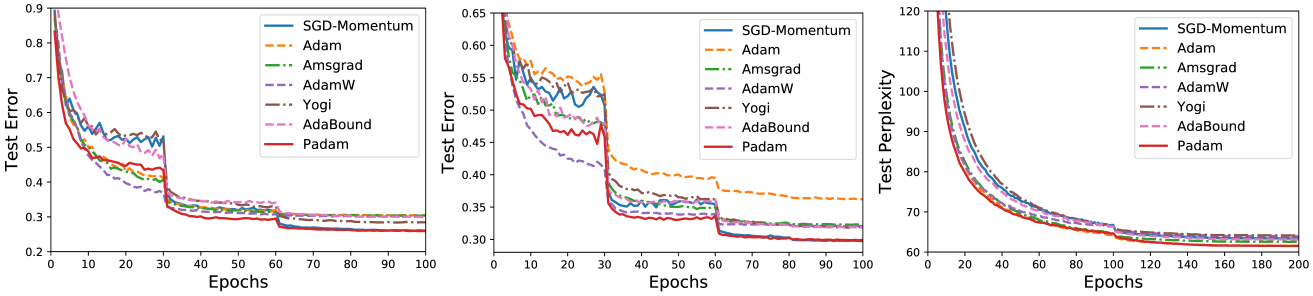


(a) Train Loss for VGGNet (b) Train Loss for ResNet (c) Train Loss for WideResNet

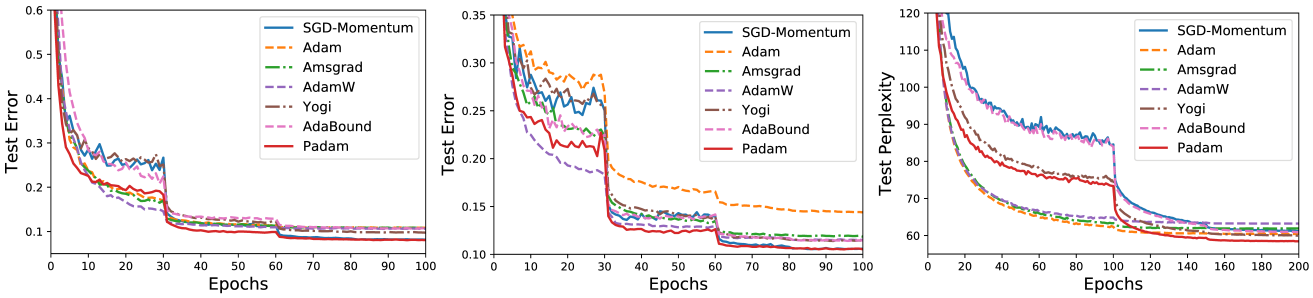


(d) Test Error for VGGNet (e) Test Error for ResNet (f) Test Error for WideResNet

Figure 2: Train loss and test error (top-1) on the CIFAR-10 dataset.



(a) Top-1 Error, VGGNet (b) Top-1 Error, ResNet (c) 2-layer LSTM



(d) Top-5 Error, VGGNet (e) Top-5 Error, ResNet (f) 3-layer LSTM

Figure 3: Test error on the ImageNet dataset (left and middle columns), and test perplexity on the Penn Treebank dataset (right column).

2-layer LSTM model and 10 for 3-layer LSTM. The momentum parameter is set to be 0.9 for both models. For all adaptive gradient methods except Padam and Yogi, we set the base learning rate as 0.001. For Yogi, we set the base learning rate as 0.01 for 2-layer LSTM model and 0.1 for 3-layer LSTM model. For Padam, we set the base learning rate as 0.01 for 2-layer LSTM model and 1 for 3-layer LSTM model. For all adaptive gradient methods, we set $\beta_1 = 0.9$, $\beta_2 = 0.999$. In terms of algorithm specific parameters, for Padam, we set the partially adaptive parameter p as 0.4 for 2-layer LSTM model

Models	SGD-Momentum	Adam	Amsgrad	AdamW	Yogi	AdaBound	Padam
VGGNet	93.71	92.21	92.54	93.54	92.94	93.28	**93.78**
ResNet	**95.00**	92.89	93.53	94.56	93.92	94.16	94.94
WideResNet	95.26	92.27	92.91	95.08	94.23	93.85	**95.34**

Table 1: Test accuracy (%) of all algorithms after 200 epochs on the CIFAR-10 dataset. Bold number indicates the best result.

Models	Test Accuracy	SGD-Momentum	Adam	Amsgrad	AdamW	Yogi	AdaBound	Padam
Resnet	Top-1	**70.23**	63.79	67.69	67.93	68.23	68.13	70.07
	Top-5	89.40	85.61	88.15	88.47	88.59	88.55	**89.47**
VGGNet	Top-1	73.93	69.52	69.61	69.89	71.56	70.00	**74.04**
	Top-5	91.82	89.12	89.19	89.35	90.25	89.27	**91.93**

Table 2: Test accuracy (%) of all algorithms after 100 epochs on the ImageNet dataset. Bold number indicates the best result.

and 0.2 for 3-layer LSTM model. For AdaBound, we set the final learning rate as 10 for 2-layer LSTM model and 100 for 3-layer LSTM model. For Yogi, ϵ is set as 10^{-3} as suggested in the original paper. For AdamW, the normalized weight decay factor is set to 4×10^{-4}. The minibatch size is set to be 20 for all LSTM experiments.

5.1 Experimental Results

We compare our proposed algorithm with other baselines on training the aforementioned three modern CNN architectures for image classification on the CIFAR-10 and ImageNet datasets. Figure 2 plots the train loss and test error (top-1 error) against training epochs on the CIFAR-10 dataset. As we can see from the figure, at the early stage of the training process, (partially) adaptive gradient methods including Padam, make rapid progress lowing both the train loss and the test error, while SGD with momentum converges relatively slowly. After the first learning rate decaying at the 100-th epoch, different algorithms start to behave differently. SGD with momentum makes a huge drop while fully adaptive gradient methods (Adam and Amsgrad) start to generalize badly. Padam, on the other hand, maintains relatively good generalization performance and also holds the lead over other algorithms. After the second decaying at the 150-th epoch, Adam and Amsgrad basically lose all the power of traversing through the parameter space due to the "small learning dilemma", while the performance of SGD with momentum finally catches up with Padam. AdamW, Yogi and AdaBound indeed improve the performance compared with original Adam but the performance is still worse than Padam. Overall we can see that Padam achieves the best of both worlds (i.e., Adam and SGD with momentum): it maintains faster convergence rate while also generalizing as well as SGD with momentum in the end.

Figure 3 (a)(b)(d)(e) plot the Top-1 and Top-5 error against training epochs on the ImageNet dataset for both VGGNet and ResNet. We can see that on the ImageNet dataset, all methods behave similarly as in our CIFAR-10 experiments. Padam method again obtains the best from both worlds by achieving the fastest convergence while generalizing as well as SGD with momentum. Even though methods such as AdamW, Yogi and AdaBound have better performance than standard Adam, they still suffer from a big generalization gap on the ImageNet dataset. Note that we did not conduct WideResNet experiment on the Imagenet dataset due to GPU memory limits.

We also perform experiments on the language modeling tasks to test our proposed algorithm on Long Short-Term Memory (LSTM) network [Hochreiter and Schmidhuber, 1997], where adaptive gradient methods such as Adam are currently the mainstream optimizers for these tasks. Figure 3 (c)(f) plot the test perplexity against training epochs on the Penn Treebank dataset [Marcus *et al.*, 1993] for both 2-layer LSTM and 3-layer LSTM models. We can observe that the differences on simpler 2-layer LSTM model is not very obvious but on more complicated 3-layer LSTM model, different algorithms have quite different optimizing behaviors. Even though Adam, Amsgrad and AdamW have faster convergence in the early stages, Padam achieves the best final test perplexity on this language modeling task for both of our experiments.

For a more quantitative comparison, we also provide the test accuracy for all above experiments. Table 1 shows the test accuracy of all algorithms on the CIFAR-10 dataset. On the CIFAR-10 dataset, methods such as Adam and Amsgrad have the lowest test accuracy. Even though more recent algorithms AdamW, Yogi, AdaBound improve upon original Adam, they still fall behind or barely match the performance of SGD with momentum. In contrast, Padam achieves the highest test accuracy for VGGNet and WideResNet on CIFAR-10 dataset. For training ResNet on the CIFAR-10 dataset, Padam is also on a par with SGD with momentum at the final epoch (differences less than 0.2%). Table 2 shows the final test accuracy of all algorithms on the ImageNet dataset. Again, we can observe that Padam achieves the best test accuracy for VGGNet (both Top-1 and Top-5) and Top-1 accuracy for ResNet. It stays very close to the best baseline of Top-1 accuracy for the ResNet model. Table 3 shows the final test perplexity of all algorithms on the Penn Treebank dataset. As we can see, Padam achieves the best (lowest) test perplexity on both 2-layer LSTM and 3-layer LSTM models. All these experimental results suggest that practitioners can use Padam for

Models	SGD-Momentum	Adam	Amsgrad	AdamW	Yogi	AdaBound	Padam
2-layer LSTM	63.37	61.58	62.56	63.93	64.13	63.14	**61.53**
3-layer LSTM	61.22	60.44	61.92	63.24	60.01	60.89	**58.48**

Table 3: Test perplexity (lower is better) of all algorithms after 200 epochs on the Penn Treebank dataset. Bold number indicates the best result.

training deep neural networks, without worrying about the generalization performances.

6 Conclusions and Future Work

In this paper, we proposed Padam, which unifies Adam/Amsgrad with SGD-Momentum. With an appropriate choice of the partially adaptive parameter, we show that Padam can achieve the best from both worlds, i.e., maintaining fast convergence rate while closing the generalization gap. We also provide a theoretical analysis towards the convergence rate of Padam to a stationary point for stochastic nonconvex optimization.

It would also be interesting to see how well Padam performs in other types of neural networks, such as generative adversarial network (GAN) [Goodfellow *et al.*, 2014] and graph convolutional neural network (GCN) [Kipf and Welling, 2017; Zou *et al.*, 2019]. We leave it as a future work.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This research was sponsored in part by the National Science Foundation CAREER Award IIS-1906169, BIGDATA IIS-1855099 and IIS-1903202. We also thank AWS for providing cloud computing credits associated with the NSF BIGDATA award. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

[Basu *et al.*, 2018] Amitabh Basu, Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for rmsprop and adam in non-convex optimization and their comparison to nesterov acceleration on autoencoders. *arXiv preprint arXiv:1807.06766*, 2018.

[Chen *et al.*, 2018] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for nonconvex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

[Davis *et al.*, 2019] Damek Davis, Dmitriy Drusvyatskiy, and Vasileios Charisopoulos. Stochastic algorithms with geometric step decay converge linearly on sharp functions. *arXiv preprint arXiv:1907.09547*, 2019.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009.

[Dozat, 2016] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[Duchi *et al.*, 2011] 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.

[Ge *et al.*, 2019] Rong Ge, Sham M Kakade, Rahul Kidambi, and Praneeth Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure. *arXiv preprint arXiv:1904.12838*, 2019.

[Ghadimi and Lan, 2013] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[Ghadimi and Lan, 2016] Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[He *et al.*, 2016] 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 (CVPR)*, pages 770–778, 2016.

[Hinton *et al.*, 2012] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, 2012.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Howard *et al.*, 2017] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[Keskar and Socher, 2017] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[Li and Orabona, 2018] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. *arXiv preprint arXiv:1805.08114*, 2018.

[Liu *et al.*, 2019] Mingrui Liu, Youssef Mroueh, Jerret Ross, Wei Zhang, Xiaodong Cui, Payel Das, and Tianbao Yang. Towards better understanding of adaptive gradient algorithms in generative adversarial nets. *arXiv preprint arXiv:1912.11940*, 2019.

[Loshchilov and Hutter, 2019] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[Luo *et al.*, 2019] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, Louisiana, May 2019.

[Marcus *et al.*, 1993] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[Mukkamala and Hein, 2017] Mahesh Chandra Mukkamala and Matthias Hein. Variants of rmsprop and adagrad with logarithmic regret bounds. In *International Conference on Machine Learning*, pages 2545–2553, 2017.

[Reddi *et al.*, 2016] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323, 2016.

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

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

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

[Ward *et al.*, 2018] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, from any initialization. *arXiv preprint arXiv:1806.01811*, 2018.

[Wilson *et al.*, 2017] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4151–4161, 2017.

[Wu *et al.*, 2018] Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. In *International Conference on Learning Representations*, 2018.

[Xie *et al.*, 2017] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995. IEEE, 2017.

[Xu *et al.*, 2016] Yi Xu, Qihang Lin, and Tianbao Yang. Accelerate stochastic subgradient method by leveraging local growth condition. *arXiv preprint arXiv:1607.01027*, 2016.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12, 2016.

[Zaheer *et al.*, 2018] Manzil Zaheer, Sashank Reddi, Devendra Singh Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems*, 2018.

[Zeiler, 2012] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[Zhang *et al.*, 2017] Zijun Zhang, Lin Ma, Zongpeng Li, and Chuan Wu. Normalized direction-preserving adam. *arXiv preprint arXiv:1709.04546*, 2017.

[Zou and Shen, 2018] Fangyu Zou and Li Shen. On the convergence of adagrad with momentum for training deep neural networks. *arXiv preprint arXiv:1808.03408*, 2018.

[Zou *et al.*, 2019] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *Advances in Neural Information Processing Systems*, pages 11247–11256, 2019.