Momentum Improves Normalized SGD

Ashok Cutkosky [1 2] **Harsh Mehta** [1]

Abstract

We provide an improved analysis of normalized SGD showing that adding momentum provably removes the need for large batch sizes on non-convex objectives. Then, we consider the case of objectives with bounded second derivative and show that in this case a small tweak to the momentum formula allows normalized SGD with momentum to find an ϵ-critical point in $O(1/\epsilon^{3.5})$ iterations, matching the best-known rates without accruing any logarithmic factors or dependence on dimension. We also provide an adaptive method that automatically improves convergence rates when the variance in the gradients is small. Finally, we show that our method is effective when employed on popular large scale tasks such as ResNet-50 and BERT pretraining, matching the performance of the disparate methods used to get state-of-the-art results on both tasks.

1. Non-Convex Stochastic Optimization

The rise of deep learning has focused research attention on the problem of solving optimization problems that are *high-dimensional*, *large-scale*, and *non-convex*. Modern neural networks can have billions of parameters (high-dimensional) (Raffel et al., 2019; Shazeer et al., 2017), are trained using datasets containing millions of examples (large scale) (Deng et al., 2009) on objective functions that are non-convex. Because of these considerations, stochastic gradient descent (SGD) has emerged as the de-facto method-of-choice for training deep models. SGD has several properties that make it attractive for this setting. First, it can operate in a streaming manner by running over the large dataset while only keeping a few examples in memory at any one time. Second, SGD has a *dimension-free* convergence guarantee for finding critical points on non-convex objectives. This means that SGD's convergence rate is independent of the dimension of the problem, allowing it to scale more easily to massive neural networks. In this paper we will investigate a variant of SGD that incorporates *gradient normalization* and *momentum*, which are both commonly used empirical modifications to SGD that are poorly understood in the non-convex setting.

A common way to analyze algorithms for training neural nets is through the lens of stochastic optimization. In this setting, we are interested in minimizing a function:

$$F(\vec{w}) = \mathop{\mathbb{E}}_{\xi\sim\mathcal{D}}[f(\vec{w},\xi)]$$

Where here $\vec{w} \in \mathbb{R}^d$ and D is a distribution over an arbitrary space Ξ. This general formulation allows for cleaner analysis, but for a more concrete intuition, $\vec{w}$ may represent the weights of a neural network, ξ may represent an individual training example, and $\mathcal{D}$ is the distribution over training examples. We do not know the entire function F, but we are allowed to access F through a *stochastic gradient oracle*. That is, given any $\vec{w}$, we may compute $\nabla f(\vec{w},\xi)$ for some ξ drawn i.i.d. from $\mathcal{D}$. Under this formulation, the SGD algorithm employs the simple iterative update:

$$\vec{w}_{t+1} = \vec{w}_t - \eta_t \nabla f(\vec{w}_t, \xi_t)$$

where $\xi_1, \ldots, \xi_T$ are i.i.d. examples from the distribution $\mathcal{D}$, and $\eta_t \in \mathbb{R}$ is a a scalar called the *learning rate*. Since our objective F is non-convex, it may be computationally intractable to find a true minimizer. Instead, we will look for an ϵ-critical point. That is, we would like our optimization algorithm to output a point $\vec{w}$ such that

$$\mathbb{E}[\|\nabla F(\vec{w})\|] \leq \epsilon$$

where $\|\cdot\|$ indicates the standard 2-norm. The expectation is taken over the randomness of the stochastic gradient oracle and any randomness inherent in the algorithm itself. In order to make the problem tractable, we make three structural assumptions. First, we assume that F is L-smooth, which means that for all $\vec{w}$ and $\vec{x}$,

$$\|\nabla F(\vec{w}) - \nabla F(\vec{x})\| \leq L\|\vec{w} - \vec{x}\| \tag{A1}$$

Second, we assume that the objective F is bounded from below, and without loss of generality (since our algorithms

[1]Google Research, California, USA [2]Boston University, Massachusetts, USA. Correspondence to: Ashok Cutkosky <ashok@cutkosky.com>, Harsh Mehta <harshm@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

will only access gradients rather than function values), we may assume it is positive:

$$F(\vec{w}) \geq 0 \tag{A2}$$

Finally, we assume that the stochastic gradient oracle has bounded variance:

$$\mathop{\mathbb{E}}_{\xi}\left[\|\nabla f(\vec{w},\xi) - \nabla F(\vec{w})\|^2\right] \leq \sigma^2 \tag{A3}$$

Under these assumptions, SGD with an optimally tuned learning rate ensures that after T iterations, we can output a point $\vec{w}$ such that (Ghadimi & Lan, 2013):

$$\mathbb{E}[\|\nabla F(\vec{w})\|] \leq O\left(\frac{1}{\sqrt{T}} + \frac{\sqrt{\sigma}}{T^{1/4}}\right) \tag{1}$$

Spurred by the empirical success of SGD, there has been a tremendous amount of work in designing modifications that improve upon its performance in various ways. However, it has recently been shown that the $O(1/T^{1/4})$ rate is optimal in the worst-case (Arjevani et al., 2019), and so improvements must either make more assumptions about the problem setting, or provide an algorithm that can somehow interpolate between a worst-case and non-worst-case problem.

One popular modification to SGD is the use of *adaptive learning rates*, popularized by AdaGrad (Duchi et al., 2010). These learning rates enable SGD to converge faster when the objective under consideration is in some technical sense "easier" than a worst-case objective[1], specifically when the variance of the loss from one example to another is small (Li & Orabona, 2019; Ward et al., 2019). The success of AdaGrad has inspired a huge number of related algorithms, including notably the Adam algorithm (Kingma & Ba, 2014).

One of the key improvements added to AdaGrad by Adam is the use of *momentum*. Inspired by the heavy-ball and acceleration algorithms in convex optimization (Polyak, 1964; Nesterov, 1983), momentum attempts to improve the convergence rate on non-convex objectives by modifying the update to have the form:

$$\vec{m}_t = \beta\vec{m}_{t-1} + (1-\beta)\nabla f(\vec{w}_t,\xi_t)$$
$$\vec{w}_{t+1} = \vec{w}_t - \eta_t\vec{m}_t$$

Intuitively, the value $\vec{m}$ is holding a running average of the past gradient values, and the hope is that this style of update may provide some kind of better stability that enables improvements over the base SGD. Momentum has had dramatic empirical success, but although prior analyses have considered momentum updates (Reddi et al., 2018; Zaheer et al., 2018), none of these have shown a strong theoretical benefit in using momentum, as their bounds do not improve on (1).

Finally, a third popular modification is the use of *normalized* updates. For example, the LARS (You et al., 2017) and LAMB (You et al., 2019) optimizers use updates similar to:

$$\vec{m}_t = \beta\vec{m}_{t-1} + (1-\beta)\nabla f(\vec{w}_t,\xi_t) \tag{2}$$

$$\vec{w}_{t+1} = \vec{w}_t - \eta_t\frac{\vec{m}_t}{\|\vec{m}_t\|} \tag{3}$$

Intuitively, this style of update attempts to capture the idea that in non-convex objectives, unlike convex ones, the *magnitude* of the gradient provides less information about the value of the function, while the *direction* still indicates the direction of steepest descent. However, all analyses of normalized updates we know of (e.g. (You et al., 2017; 2019; Hazan et al., 2015)) require the variance of the gradient oracle to be very small, or, equivalently, for the algorithm to make use of an extremely large batch-size in order to achieve any convergence guarantees. Intuitively, this requirement arises because the normalization can inflate very small errors: even if $m_t = \nabla F(\vec{w}_t) + \zeta$ for some small error ζ, it might be that $\frac{m_t}{\|m_t\|}$ is actually very far from $\frac{\nabla F(\vec{w}_t)}{\|\nabla F(\vec{w}_t)\|}$. Nevertheless, this update has also been used to empirically accelerate training neural networks.

From a more theoretical perspective, a recent approach to going beyond the SGD rate (1) is to add some additional structural assumptions to the loss. One appealing assumption is *second-order smoothness*, in which we assume that the third derivative of F has bounded operator norm. Equivalently, for all $\vec{w}$, and $\vec{z}$, we have

$$\|\nabla^2 F(\vec{w}) - \nabla^2 F(\vec{z})\|_{\text{op}} \leq \rho\|\vec{w} - \vec{z}\| \tag{A4}$$

Many previous works have achieved improved results by utilizing this assumption in concert with stronger oracles, such as evaluating two gradients per each example or evaluating Hessian-vector products (Allen-Zhu, 2018; Tripuraneni et al., 2018). However, using our same stochastic gradient oracle model and this second-order smoothness assumption, it was recently proven (Fang et al., 2019) that a variant of SGD can achieve a convergence rate of

$$\|\nabla F(\vec{w})\| \leq O\left(\frac{\text{polylog(d)}}{T^{2/7}}\right)$$

where d is the dimension of the problem.[2] This breakthrough result shows that even for fairly high-dimensional

[1] In this paper we use "adaptive" in a statistical sense of the word: an algorithm is adaptive if it automatically adjusts itself to some unknown parameter, such as the variance of the gradients. This is different from the idea of using a different learning rate for each dimension of an optimization problem that was also popularized by (Duchi et al., 2010).

[2] In fact, this result provided a convergence to a *local minimum*, which is stronger than a critical point.

problems, SGD can obtain faster convergence than the initial analysis suggests. However, in modern deep learning architectures, d can be on the order of billions (Radford et al., 2019). In this regime, it may easily hold that the logarithmic term is large enough that this new analysis of SGD does not actually suggest improved performance over the previous $O(1/T^{1/4})$ rate. To deal with extremely high-dimensional regimes, we would like a convergence rate that is completely dimension-free.

In a somewhat orthogonal direction, several algorithms have been proposed based on *variance reduction* (Johnson & Zhang, 2013). Recently, these have provided provable improvements over SGD's convergence rate, finding a point with $\mathbb{E}[\|\nabla F(\vec{w})\|] \leq O(1/T^{1/3})$ after T iterations (Fang et al., 2018; Zhou et al., 2018). Unfortunately, these algorithms require *two* gradient evaluations for every i.i.d. sample ξ. Our model requires each ξ to be used only once, and so forbids this operation. Moreover, it turns out to be surprisingly hard to implement this two-gradient model efficiently in modern machine learning frameworks. Nevertheless, more recent work in this area has produced algorithms whose updates are very similar to the (unnormalized) momentum update (Cutkosky & Orabona, 2019; Tran-Dinh et al., 2019), suggesting that momentum may operate by reducing variance in SGD.

Finally, another technique that also suggests a connection to momentum in reducing variance is *implicit gradient transport* (Arnold et al., 2019). Implicit gradient transport is a very recent discovery that provably reduces the variance in gradient estimates in the special case that the Hessian of the objective is *constant* (e.g. if the objective is a linear regression problem). The original presentation considers the following version of momentum:

$$\vec{m}_t = \frac{t}{t+1}\vec{m}_{t-1} + \frac{1}{t+1}\nabla f(\vec{w}_t + t(\vec{w}_t - \vec{w}_{t-1}), \xi_t)$$
$$\vec{w}_{t+1} = \vec{w}_t - \eta_t \vec{m}_t$$

To gain some intuition for why this is a good idea when the Hessian is constant, suppose we have $\vec{m}_{t-1} = \nabla F(\vec{w}_{t-1}) + X$ where X is some mean-zero random variable with variance σ^2/t. Let us also write $\nabla f(\vec{w}_t + t(\vec{w}_t - \vec{w}_{t-1}), \xi_t) = \nabla F(\vec{w}_t + t(\vec{w}_t - \vec{w}_{t-1})) + Y$ where Y is also a mean-zero random variable with variance σ^2. Then since the Hessian is constant, we have:

$$\nabla F(\vec{w}_{t-1}) = \nabla F(\vec{w}_t) + \nabla^2 F(\vec{w}_{t-1} - \vec{w}_t)$$
$$\nabla F(\vec{w}_t + t(\vec{w}_t - \vec{w}_{t-1})) = t\nabla^2 F(\vec{w}_t - \vec{w}_{t-1}) + \nabla F(\vec{w}_t)$$
$$\vec{m}_t = \nabla F(\vec{w}_t) + \frac{tX + Y}{t+1}$$

Notice that $\frac{tX+Y}{t+1}$ has variance $\sigma^2/(t+1)$, so that by induction, we will have for all t that $\vec{m}_t$ is an unbiased estimate of $\nabla F(\vec{w}_t)$ with variance only $\sigma^2/(t+1)$. This technique shows great promise, but the current theoretical analysis is limited to convex quadratic objectives.

In this paper we address some of these preceding issues. First, we show that the normalized stochastic gradient descent update with momentum does *not* require small variance or large batches in order to match the convergence rate of SGD. Next, we tackle the case of second-order smooth objectives. For this, we introduce a further modification of the momentum update based on the implicit gradient transport idea. We show that combining this newer kind of momentum with normalized gradient descent enables a dimension-free convergence rate of $O(1/T^{2/7})$. Moreover, we feel that our new analysis is substantially more straightforward than prior work, as demonstrated by the number of pages required to prove our results. Finally, we propose an adaptive version of our algorithm and show that this final algorithm's convergence guarantee automatically improves when the stochastic gradient oracle has small variance. We hope our results can shed some light in the empirical success of momentum in training deep networks.

In addition to the theoretical discussion, we also demonstrate effectiveness of our method, `NIGT` (pronounced "night"), on popular deep learning tasks. We show comparisons of our method on 1) BERT pretraining and 2) ImageNet classification with Resnet-50. Adam is typically used to achieve state of the art results on BERT task (Devlin et al., 2019) whereas SGD is used for Resnet-50 on Imagenet dataset since Adam fails to perform well on it (You et al., 2019; Anil et al., 2019). We show comparison with the stronger baseline in each case. Finally, since momentum is the only significant slot variable kept, our method incurs greatly reduced memory overhead compared to other methods such as Adam. This is a significant practical advantage due to the continuing trend of increasingly large models (Shazeer & Stern, 2018; Anil et al., 2019).

The rest of this paper is organized as follows: in Section 2, we provide our first result showing that momentum can be used to "rescue" normalized SGD in the high-variance/small-batch regime. In Section 3, we expand upon this theory to show that incorporating a momentum update inspired by (Arnold et al., 2019) allows for improved convergence rates when the objective is second-order smooth, and we proceed to incorporate adaptivity into our analysis in Section 4. In Section 5 we describe our implementation and experimental study, and we provide some concluding remarks and open problems in Section 6.

2. Normalized SGD with Momentum

In this section, we analyze the normalized SGD with momentum update given by equations (2) and (3). The result is Theorem 1, which shows that the addition of momentum

allows normalized SGD to match the optimal convergence rate obtained by ordinary SGD, without requiring any large batch sizes. To gain some intuition for why the momentum is necessary in this case, consider a one-dimensional optimization scenario in which $\nabla f(\vec{w}, \xi) = p$ with probability $1-p$ and $p-1$ with probability p for some $p \in (0, 1/2)$. Then we clearly have $\nabla F(\vec{w}) = \mathbb{E}[\nabla f(\vec{w}, \xi)] = 0$, so that intuitively the "correct" thing for the algorithm to do is to not move, at least on average. Unfortunately, we have $\mathbb{E}\left[\frac{\nabla f(\vec{w},\xi)}{\|\nabla f(\vec{w},\xi)\|}\right] = 1 - 2p > 0$, so that the algorithm will be biased to moving *away* from this critical point. The problem here was that, without momentum, the error in the gradient estimate before normalization is much larger than the true value of the gradient. In order to fix this, one must decrease the variance in the stochastic gradient oracle by employing a very large batch size, usually on the order of T (You et al., 2019; Hazan et al., 2015). Our Theorem 1 avoids this requirement and matches the optimal bound (1) up to constants.

Theorem 1. *Suppose F and $\mathcal{D}$ satisfy the assumptions (A1), (A2) and (A3). Let $\vec{w}_1$ be some given initial point and set $\vec{m}_1 = \nabla f(\vec{w}_1, \xi_1)$. Let R be any upper bound on $F(\vec{w}_1)$. Set $\alpha = \min\left(\frac{\sqrt{RL}}{\sigma\sqrt{T}}, 1\right)$ and $\eta = \frac{\sqrt{R\alpha}}{\sqrt{TL}}$ Then let $\vec{w}_t$ be the iterates produced by the recurrences (2) and (3) with $\eta_t = \eta$ and $\beta_t = 1 - \alpha$ for all t. Then the average norm of $\|\nabla F(\vec{w}_t)\|$ satisfies:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla F(\vec{w}_t)\|\right] \leq \frac{29\sqrt{RL}}{\sqrt{T}} + \frac{21\sqrt{\sigma}(RL)^{1/4}}{T^{1/4}} + \frac{8\sigma}{\sqrt{RLT}}$$

Before proving this Theorem, we provide a general technical Lemma about each step of normalized SGD:

Lemma 2. *Suppose F and $\mathcal{D}$ satisfy the assumptions (A1) and (A3). Let $\vec{w}_1$ be some initial point, and consider the updates:*

$$\vec{w}_{t+1} = \vec{w}_t - \eta_t \frac{\hat{g}_t}{\|\hat{g}_t\|}$$

where $\hat{g}_t$ is generated by some arbitrary process or algorithm. Let $\hat{\epsilon}_t = \hat{g}_t - \nabla F(\vec{w}_t)$. Then we have

$$F(\vec{w}_{t+1}) - F(\vec{w}_t) \leq -\frac{\eta_t}{3}\|\nabla F(\vec{w}_t)\| + \frac{8\eta_t}{3}\|\hat{\epsilon}_t\| + \frac{L\eta_t^2}{2}$$

In particular, if F also satisfies (A2) and η_t is a constant η for all t, we have:

$$\sum_{t=1}^{T}\|\nabla F(\vec{w}_t)\| \leq \frac{3F(\vec{w}_1)}{\eta} + \frac{3LT\eta}{2} + 8\sum_{t=1}^{T}\|\hat{\epsilon}_t\|$$

Proof. Since F is L-smooth, we have

$$\begin{aligned} F(\vec{w}_{t+1}) - F(\vec{w}_t) &\leq \\ &\langle \nabla F(\vec{w}_t), \vec{w}_{t+1} - \vec{w}_t\rangle + \frac{L\|\vec{w}_{t+1} - \vec{w}_t\|^2}{2} \\ &= -\eta_t \frac{\langle \nabla F(\vec{w}_t), \hat{g}_t\rangle}{\|\hat{g}_t\|} + \frac{L\eta_t^2}{2} \end{aligned}$$

Now consider two cases, either $\|\nabla F(\vec{w}_t)\| \geq 2\|\hat{\epsilon}_t\|$ or not. In the former case, we have

$$\begin{aligned} -\frac{\langle \hat{g}_t, \nabla F(\vec{w}_t)\rangle}{\|\hat{g}_t\|} &\leq -\frac{\|\nabla F(\vec{w}_t)\|^2 + \langle \nabla F(\vec{w}_t), \hat{\epsilon}_t\rangle}{\|\nabla F(\vec{w}_t) + \hat{\epsilon}_t\|} \\ &\leq -\frac{\|\nabla F(\vec{w}_t)\|^2}{2\|\nabla F(\vec{w}_t) + \hat{\epsilon}_t\|} \\ &\leq -\frac{\|\nabla F(\vec{w}_t)\|}{3} \\ &\leq -\frac{\|\nabla F(\vec{w}_t)\|}{3} + \frac{8\|\hat{\epsilon}_t\|}{3} \end{aligned}$$

In the latter case, we have

$$\begin{aligned} -\frac{\langle \hat{g}_t, \nabla F(\vec{w}_t)\rangle}{\|\hat{g}_t\|} &\leq \|\nabla F(\vec{w}_t)\| \\ &= -\frac{\|\nabla F(\vec{w}_t)\|}{3} + \frac{4\|\nabla F(\vec{w}_t)\|}{3} \\ &\leq -\frac{\|\nabla F(\vec{w}_t)\|}{3} + \frac{8\|\hat{\epsilon}_t\|}{3} \end{aligned}$$

Thus, the same bound holds in both cases. Putting everything together now proves the first statement of the Lemma. The second statement follows by summing over T, observing that the left-hand-side of the equation telescopes, and rearranging. □

By this Lemma, if we can show that our choices for β and η result in a small value for $\sum_{t=1}^{T}\|\hat{\epsilon}_t\|$, then setting η appropriately will prove Theorem 1. We carry out this agenda in more detail below:

Proof of Theorem 1. Define $\hat{\epsilon}_t = \vec{m}_t - \nabla F(\vec{w}_t)$ and define $\epsilon_t = \nabla f(\vec{w}_t, \xi_t) - \nabla F(\vec{w}_t)$. Notice that by our assumptions, we have

$$\begin{aligned} \mathbb{E}[\|\epsilon_t\|^2] &\leq \sigma^2 \\ \mathbb{E}[\langle \epsilon_i, \epsilon_j\rangle] &= 0 \text{ for } i \neq j \end{aligned}$$

Further, define $S(a, b) = \nabla F(a) - \nabla F(b)$. By smoothness, we must have $\|S(\vec{w}_t, \vec{w}_{t+1})\| \leq L\|\vec{w}_t - \vec{w}_{t+1}\| = \eta L$ for all t. With this notation, we have the following recursive formulation for any $t \geq 1$:

$$\begin{aligned} \vec{m}_{t+1} &= (1-\alpha)(\nabla F(\vec{w}_t) + \hat{\epsilon}_t) + \alpha \nabla f(\vec{w}_{t+1}, \xi_{t+1}) \\ &= \nabla F(\vec{w}_{t+1}) + (1-\alpha)(S(\vec{w}_t, \vec{w}_{t+1}) + \hat{\epsilon}_t) + \alpha\epsilon_{t+1} \\ \hat{\epsilon}_{t+1} &= (1-\alpha)S(\vec{w}_t, \vec{w}_{t+1}) + (1-\alpha)\hat{\epsilon}_t + \alpha\epsilon_{t+1} \end{aligned}$$

Now, we unravel the recursion for t iterations:

$$\hat{\epsilon}_{t+1} = (1-\alpha)^t \hat{\epsilon}_1 + \alpha \sum_{\tau=0}^{t-1} (1-\alpha)^\tau \epsilon_{t+1-\tau} + (1-\alpha) \sum_{\tau=0}^{t-1} (1-\alpha)^\tau S(\vec{w}_{t-\tau}, \vec{w}_{t+1-\tau})$$

Next, take the magnitude of both sides and use triangle inequality:

$$\|\hat{\epsilon}_{t+1}\| \le (1-\alpha)^t \|\epsilon_1\| + \alpha \left\| \sum_{\tau=0}^{t-1} (1-\alpha)^\tau \epsilon_{t+1-\tau} \right\| + (1-\alpha)\eta L \sum_{\tau=0}^{t-1} (1-\alpha)^\tau$$

Where in the above we have observed that $\hat{\epsilon}_1 = \epsilon_1$. Now take expectation, and use the fact that any two ϵ_i are uncorrelated and apply Jensen inequality to obtain:

$$\begin{aligned}
\mathbb{E}\left[\|\hat{\epsilon}_{t+1}\|\right] &\le (1-\alpha)^t \mathbb{E}\left[\|\epsilon_1\|\right] + \alpha \sqrt{\sum_{\tau=0}^{t-1} (1-\alpha)^{2\tau} \sigma^2} \\
&\quad + (1-\alpha)\eta L \sum_{\tau=0}^{t-1} (1-\alpha)^\tau \\
&= (1-\alpha)^t \sigma + \frac{\alpha\sigma}{\sqrt{1-(1-\alpha)^2}} + \frac{\eta L}{\alpha} \\
&\le (1-\alpha)^t \sigma + \sqrt{\alpha}\sigma + \frac{\eta L}{\alpha} \\
\mathbb{E}\left[\sum_{t=1}^{T} \|\hat{\epsilon}_t\|\right] &\le \frac{\sigma}{\alpha} + T\sqrt{\alpha}\sigma + \frac{T\eta L}{\alpha}
\end{aligned}$$

Applying Lemma 2 now yields:

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla F(\vec{w}_t)\|\right] &\le \frac{3F(\vec{w}_1)}{\eta} + \frac{3LT\eta}{2} + 8\sum_{t=1}^{T} \|\hat{\epsilon}_t\| \\
&\le \frac{3R}{\eta} + \frac{3TL\eta}{2} + \frac{8\sigma}{\alpha} \\
&\quad + 8T\sqrt{\alpha}\sigma + \frac{8TL\eta}{\alpha} \\
&\le \frac{3R}{\eta} + \frac{8\sigma}{\alpha} + 8T\sqrt{\alpha}\sigma + \frac{10TL\eta}{\alpha}
\end{aligned}$$

Where we have used $\alpha \le 1$. Set $\alpha = \min\left(\frac{\sqrt{RL}}{\sigma\sqrt{T}}, 1\right)$ and $\eta = \frac{\sqrt{R\alpha}}{\sqrt{TL}}$. Observe from the setting of η that we have:

$$\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla F(\vec{w}_t)\|\right] \le \frac{13\sqrt{RLT}}{\sqrt{\alpha}} + 8\sqrt{\alpha}\sigma T + \frac{8\sigma}{\alpha}$$

Now suppose $\alpha = 1$. This implies that $\sigma \le \frac{\sqrt{RL}}{\sqrt{T}}$. Therefore the RHS of the above expression is bounded by $29\sqrt{RLT}$. On the other hand, if $\alpha = \frac{\sqrt{RL}}{\sigma\sqrt{T}}$, the RHS is bounded by

$$21\sqrt{\sigma}(RL)^{1/4}T^{3/4} + \frac{8\sigma\sqrt{T}}{\sqrt{RL}}$$

Adding these expressions yields the desired result. □

3. Faster Convergence with Second-Order Smoothness

Now that we have seen a simpler example of how to analyze the convergence of normalized SGD with momentum in the non-convex setting, we can proceed to consider the more advanced case that F is second-order smooth. Notice that in the proof of Theorem 1, we made use of the quantity $S(a,b) = \nabla F(a) - \nabla F(b)$, which satisfies $\|S(a,b)\| \le L\|a-b\|$. In this section, we will need a related quantity:

$$Z(a,b) = \nabla F(a) - \nabla F(b) - \nabla^2 F(b)(a-b)$$

Assuming F satisfies (A4), Taylor's theorem implies that Z satisfies $\|Z(a,b)\| \le \rho\|a-b\|^2$. The improved dependency on $\|a-b\|$ from linear to quadratic will allow us to achieve a faster convergence rate.

Without further ado, we present our algorithm NIGT in Algorithm 1 below: The auxiliary variable $\vec{x}_t$ is introduced

Algorithm 1 Normalized SGD with Implicit Gradient Transport (NIGT - pronounced "night")

Input: Initial Point $\vec{w}_1$, learning rate η, momentum parameter β.
Set $\vec{m}_1 = \nabla f(\vec{x}_1, \xi_1)$.
Set $\vec{w}_2 = \vec{w}_1 - \eta \frac{\vec{m}_1}{\|\vec{m}_1\|}$.
for $t = 2 \dots T$ **do**
  Set $\vec{x}_t = \vec{w}_t + \frac{\beta}{1-\beta}(\vec{w}_t - \vec{w}_{t-1})$.
  Set $\vec{m}_t = \beta_t \vec{m}_{t-1} + (1-\beta_t)\nabla f(\vec{x}_t, \xi_t)$.
  Set $\vec{w}_{t+1} = \vec{w}_t - \eta \frac{\vec{m}_t}{\|\vec{m}_t\|}$.
end for

for notational convenience - it can be recomputed every iteration from $\vec{w}_t$ and $\vec{w}_{t-1}$. Notice that, with $\beta_t = \frac{t}{t+1}$, the update for $\vec{m}_t$ corresponds exactly to the implicit gradient transport update proposed by (Arnold et al., 2019). We will instead use the different setting $\beta = 1 - O(T^{-4/7})$. We show that this value of β, combined with the normalized SGD update, enables faster convergence on second-order smooth non-convex objectives. This significantly extends the prior results of (Arnold et al., 2019), which hold only on convex quadratic objectives with constant Hessian, and helps realize the theoretical potential of the implicit gradient transport idea.

Theorem 3. *Suppose f and $\mathcal{D}$ satisfies the assumptions (A1), (A2), (A3), and (A4). Let $\vec{w}_1$ be an arbitrary*

starting point, and set $\vec{m}_1 = \nabla f(\vec{w}_1, \xi_1)$. *Let* $R \geq F(\vec{w}_1)$ *and set* $\eta = \min\left(\frac{R^{5/7}}{T^{5/7}\rho^{1/7}\sigma^{4/7}}, \sqrt{\frac{R}{TL}}\right)$ *and* $\alpha = \min\left(\frac{R^{4/7}\rho^{2/7}}{T^{4/7}\sigma^{6/7}}, 1\right)$. *Suppose* $\vec{w}_1, \ldots, \vec{w}_T$ *are the iterates produced by Algorithm 1 with* $\eta_t = \eta$ *and* $\beta_t = 1 - \alpha$ *for all* t. *Then, the average expected gradient norm satisfies:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla F(\vec{w}_t)\|\right] \leq \frac{5\sqrt{RL}}{\sqrt{T}} + \frac{8\sigma^{13/7}}{R^{4/7}\rho^{2/7}T^{3/7}} + \frac{27R^{2/7}\rho^{1/7}\sigma^{4/7}}{T^{2/7}}$$

In particular, the dependence on T *is* $O(1/T^{2/7})$.

Proof. Our proof begins very similarly to that of Theorem 1. We define $\hat{\epsilon}_t = m_t - \nabla F(\vec{w}_t)$, and $\epsilon_t = \nabla f(\vec{x}_t, \xi_t) - \nabla F(\vec{x}_t)$. Then we have the following recursive formulation:

$$\begin{aligned}
\vec{m}_{t+1} &= (1-\alpha)(\nabla F(\vec{w}_t) + \hat{\epsilon}_t) + \alpha\nabla f(\vec{x}_{t+1}, \xi_{t+1}) \\
&= (1-\alpha)(\nabla F(\vec{w}_{t+1}) + \nabla^2 F(\vec{w}_{t+1})(\vec{w}_t - \vec{w}_{t+1})) \\
&\quad + (1-\alpha)(Z(\vec{w}_t, \vec{w}_{t+1}) + \hat{\epsilon}_t) + \alpha\nabla F(\vec{w}_{t+1}) \\
&\quad + \alpha\left(\frac{1-\alpha}{\alpha}\nabla^2 F(\vec{w}_{t+1})(\vec{w}_{t+1} - \vec{w}_t)\right) \\
&\quad + \alpha(Z(\vec{x}_{t+1}, \vec{w}_{t+1}) + \epsilon_{t+1}) \\
&= \nabla F(\vec{w}_{t+1}) + (1-\alpha)Z(\vec{w}_t, \vec{w}_{t+1}) \\
&\quad + \alpha Z(\vec{x}_{t+1}, \vec{w}_{t+1}) + (1-\alpha)\hat{\epsilon}_t + \alpha\epsilon_{t+1} \\
\hat{\epsilon}_{t+1} &= (1-\alpha)\hat{\epsilon}_t + (1-\alpha)Z(\vec{w}_t, \vec{w}_{t+1}) \\
&\quad + \alpha Z(\vec{x}_{t+1}, \vec{w}_{t+1}) + \alpha\epsilon_{t+1}
\end{aligned}$$

Here we have already used the key insight of implicit gradient transport, as well as our second-order smoothness assumption. The carefully constructed equation for $\vec{x}_t$ is such that the $\nabla^2 F(\vec{w}_{t+1})$ terms cancel out, and we keep track of the error introduced by having a non-constant Hessian in the Z function. Next, we unroll the recursion to obtain:

$$\begin{aligned}
\hat{\epsilon}_{t+1} = (1-\alpha)^t\hat{\epsilon}_1 &+ \alpha\sum_{\tau=0}^{t-1}(1-\alpha)^\tau\epsilon_{t+1-\tau} \\
&+ (1-\alpha)\sum_{\tau=0}^{t-1}(1-\alpha)^\tau Z(\vec{w}_{t-\tau}, \vec{w}_{t+1-\tau}) \\
&+ \alpha\sum_{\tau=0}^{t-1}(1-\alpha)^\tau Z(\vec{x}_{t+1}, \vec{w}_{t+1})
\end{aligned}$$

Now, recall that $Z(a,b) \leq \rho\|a-b\|$. This implies:

$$\alpha\|Z(\vec{x}_t, \vec{w}_t)\| \leq \rho\frac{(1-\alpha)^2\eta^2}{\alpha} \leq \rho\frac{(1-\alpha)\eta^2}{\alpha}$$
$$(1-\alpha)\|Z(\vec{w}_t, \vec{w}_{t+1})\| \leq (1-\alpha)\rho\eta^2 \leq \rho\frac{(1-\alpha)\eta^2}{\alpha}$$

Now just as in the proof of Theorem 1, we take magnitudes, use triangle inequality, and take expectations, but this time we use the bounds on Z:

$$\begin{aligned}
\mathbb{E}[\|\hat{\epsilon}_{t+1}\|] &\leq (1-\alpha)^t\sigma + \alpha\sqrt{\sum_{\tau=0}^{t-1}(1-\alpha)^{2\tau}\sigma^2} \\
&\quad + 2\frac{\eta^2\rho}{\alpha}\sum_{\tau=0}^{t-1}(1-\alpha)^{\tau+1} \\
&\leq (1-\alpha)^t\sigma + \frac{\alpha\sigma}{\sqrt{1-(1-\alpha)^2}} + 2\frac{\eta^2\rho(1-\alpha)}{\alpha^2} \\
&\leq (1-\alpha)^t\sigma + \sqrt{\alpha}\sigma + 2\frac{\eta^2\rho(1-\alpha)}{\alpha^2}
\end{aligned}$$

Where we have used $\hat{\epsilon}_1 = \epsilon_1$ in the first line. Now sum over t to obtain:

$$\sum_{t=1}^{T}\mathbb{E}[\|\hat{\epsilon}_t\|] \leq \frac{\sigma(1-\alpha^T)}{\alpha} + T\sqrt{\alpha}\sigma + 2\frac{T\eta^2\rho(1-\alpha)}{\alpha^2}$$

Next, we apply Lemma 2:

$$\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\vec{w}_t)\|] \leq \frac{3F(\vec{w}_1)}{\eta} + \frac{3LT\eta}{2} + \frac{8\sigma(1-\alpha^T)}{\alpha} + 8T\sqrt{\alpha}\sigma + 16\frac{T\eta^2\rho(1-\alpha)}{\alpha^2}$$

Now set $\eta = \min\left(\frac{R^{5/7}}{T^{5/7}\rho^{1/7}\sigma^{4/7}}, \sqrt{\frac{R}{TL}}\right)$, so that we have:

$$\frac{3F(\vec{w}_1)}{\eta} + \frac{3LT\eta}{2} \leq 5\sqrt{RLT} + 3R^{2/7}\rho^{1/7}\sigma^{4/7}T^{5/7}$$

Further, set $\alpha = \min\left(\frac{R^{4/7}\rho^{2/7}}{T^{4/7}\sigma^{6/7}}, 1\right)$. Notice that in all cases we have $T\sqrt{\alpha}\sigma \leq R^{2/7}\rho^{1/7}\sigma^{4/7}T^{5/7}$, so we can conclude,

$$\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\vec{w}_t)\|] \leq 5\sqrt{RLT} + 11R^{2/7}\rho^{1/7}\sigma^{4/7}T^{5/7} + \frac{8\sigma(1-\alpha^T)}{\alpha} + 16\frac{T\eta^2\rho(1-\alpha)}{\alpha^2} \quad (4)$$

Let us consider the case that $\alpha = 1$. In this case, the last two terms in (4) are zero, so we have:

$$\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\vec{w}_t)\|] \leq 5\sqrt{RLT} + 11R^{2/7}\rho^{1/7}\sigma^{4/7}T^{5/7} \quad (5)$$

Next, suppose $\alpha = \frac{R^{4/7}\rho^{2/7}}{T^{4/7}\sigma^{6/7}}$. Then instead we use the fact

that $\eta = \frac{R^{5/7}}{T^{5/7}\rho^{1/7}\sigma^{4/7}}$ to refine (4) as:

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla F(\vec{w}_t)\|] \leq$$
$$5\sqrt{RLT} + 11R^{2/7}\rho^{1/7}\sigma^{4/7}T^{5/7} + \frac{8\sigma}{\alpha} + 16\frac{T\eta^2\rho}{\alpha^2}$$
$$= 5\sqrt{RLT} + 27R^{2/7}\rho^{1/7}\sigma^{4/7}T^{5/7} + \frac{8\sigma^{13/7}T^{4/7}}{R^{4/7}\rho^{2/7}} \quad (6)$$

Taking the maximum of (5) and (6) yields the desired convergence guarantee. □

4. Adaptive Algorithm

In the previous sections, we considered the case of *constant* learning rate η and momentum β parameters. We showed that with appropriate settings, we can obtain fast convergence rates. In this section, we go further and consider varying η and β values. In particular, we will provide an *adaptive* schedule in which η and β are adjusted on-the-fly based on the observed gradients. This adaptive schedule yields a convergence guarantee that automatically improves when σ is small, *without knowing* σ. In order to do this, we will need to sample *two* gradients at each point $\vec{x}_t$. The two gradient samples are independent, so this still fits within our stochastic gradient oracle model of computation. We denote the second independent gradient estimate evaluated at the point $\vec{x}_t$ as $\nabla f(\vec{x}_t, \xi_t')$. Finally, we will add another assumption: the function F (or at least the values of $F(\vec{x}_t)$) should be bounded above by some constant M. With this notation, our adaptive algorithm is presented in Algorithm 2 and analyzed in Theorem 4 below.

Algorithm 2 Adaptive Normalized SGD with Momentum

Input: Initial point $\vec{w}_1$, lipschitz bound $\mathfrak{g}$.
Set $C = \sqrt{\frac{7}{26\mathfrak{g}^{6/7}}}$ and $D = C^{-14/3}$
Set $\vec{w}_0 = \vec{w}_1$.
Set $\vec{m}_0 = 0$.
Set $G_1 = 3\mathfrak{g}^2 + D$.
Set $\eta_0 = \frac{C}{D^{2/7}}$.
for $t = 1 \dots T$ **do**
  Set $\eta_t = \frac{C}{(G_t^2(t+1)^3)^{1/7}}$.
  Set $\alpha_t = \frac{1}{t\eta_{t-1}^2 G_{t-1}}$.
  Set $\beta_t = 1 - \alpha_t$.
  Set $\vec{x}_t = \vec{w}_t + \frac{\beta_t}{1-\beta_t}(\vec{w}_t - \vec{w}_{t-1})$.
  Set $\vec{m}_t = \beta_t \vec{m}_{t-1} + (1-\beta_t)\nabla f(\vec{x}_t, \xi_t)$.
  Set $G_{t+1} = G_t + \|\nabla f(\vec{x}_t, \xi_t) - \nabla f(\vec{x}_t, \xi_t')\|^2 + \mathfrak{g}^2((t+1)^{1/4} - t^{1/4})$.
  Set $\vec{w}_{t+1} = \vec{w}_t - \eta_t \frac{\vec{m}_t}{\|\vec{m}_t\|}$.
end for

Theorem 4. *Suppose that f and $\mathcal{D}$ satisfies (A1), (A2), (A3), and (A4). Further, suppose $\|\nabla f(\vec{w}, \xi)\| \leq \mathfrak{g}$ for all $\vec{w}$ with probability 1, and that $F(\vec{w}) \leq M$ for all $\vec{w}$ for some M. Then Algorithm 2 guarantees:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla F(\vec{w}_t)\|\right] \leq \tilde{O}\left(\frac{1}{\sqrt{T}} + \frac{\sigma^{4/7}}{T^{2/7}}\right)$$

where the $\tilde{O}$ notation hides constants that depend on R, G, and M and a factor of $\log(T)$.

We defer the proof to Appendix A.

5. Experiments

Now, we turn to experimental evaluation of the proposed method NIGT on two popular large-scale deep learning benchmarks: BERT pretraining and ResNet-50. First we explain the setup and choice of hyperparameters for our method, and then elucidate both tasks and the details on the baseline used for each task.

5.1. Setup

We implemented our algorithm in the Tensorflow framework. For simplicity, we implemented a per-layer version of our algorithm, normalizing the gradients for each layer in the network, rather than normalizing the full gradient. Taking our cue from defaults from previous empirical literature on momentum, we set the β parameter to 0.9 for NIGT for both BERT and ResNet-50. For BERT, we stick with the learning rate schedule used for Adam in (Devlin et al., 2019) i.e linear warmup and polynomial decay of $\eta_t = \eta_0 * (1 - \frac{t}{T})$. Whereas for ResNet-50, we found that linear warmup and polynomical decay of $\eta_t = \eta_0 * (1 - \frac{t}{T})^2$ worked best (You et al., 2017). We performed a grid search on base learning rate $\eta_0 \in [10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}]$ for both the tasks. In our implementation, we also scale the learning rate with the norm of the weights for each layer similar to (You et al., 2017). We did not normalize gradients for bias, batch normalization and layer normalization parameters, and we scaled their learning rates by a factor of 1000. All our experiments were conducted on a TPUv3 architecture.

5.2. BERT pretraining

We replicate most of the setup created by (Devlin et al., 2019) for our BERT baseline. Our text corpus is a concatenation of Wikipedia and BooksCorpus, with a total of roughly 3.3B words. We use examples of sequence length 512 with token masking rate of 0.15 and train the BERT-Base model with Masked Language Modeling target accuracy of 70.0% for 500k steps. Fig 1 compares masked language modeling accuracy resulting with our method vs the baseline. We set NIGT base learning rate η_0 to 0.001 and batch size to 256. We replicate the hyperparameters used for our baseline method Adam exactly from (Devlin et al., 2019) i.e. step

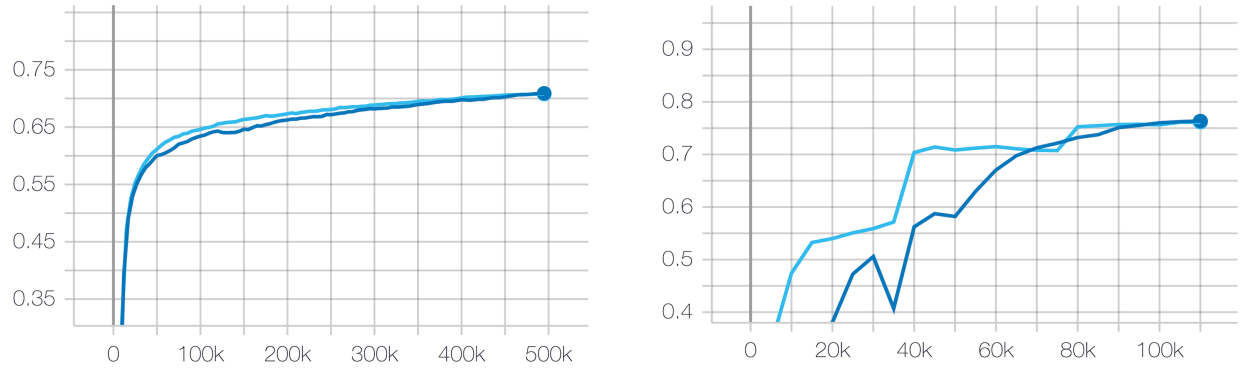

Figure 1. (a) Masked language modeling validation accuracy comparisons for our method compared to Adam. Our method NIGT (dark blue), in the end, fares slightly better with 70.91 vs 70.76 for Adam (light blue). (b) Top-1 validation accuracy attained by Resnet-50 architecture over Imagenet dataset. Accuracy at the end of 90 epochs: NIGT (dark blue) - 76.37% and SGD (light blue) - 76.2%.

size $\alpha = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$.

5.3. Resnet-50

We train the ImageNet dataset (Deng et al., 2009) with Resnet-50 architecture and compare our method against the commonly used SGD optimizer with momentum in Fig 1. Others have observed that Adam/Adagrad fails to achieve accuracy attained by SGD on this task (You et al., 2019; Anil et al., 2019). For our experiments, we set the batch size to 1024 and train the models for 90 epochs. We set the base learning rate η_0 for NIGT to 0.01. For the SGD baseline, we stick with the common practice of employing gradual warmup for 5 epochs up to a base learning rate of 0.4 and then divide it by 10 at 30-th, 60-th and 80-th epoch, as done in (Goyal et al., 2017; He et al., 2015).

6. Conclusion and Future Work

We have presented a new analysis of the effects of momentum on normalized SGD, and showed that when the objective is second-order smooth, normalized SGD with momentum can find an ϵ-critical point in $O(1/\epsilon^{3.5})$ iterations, matching the best-known rates in a dimension-free manner and surprisingly straightforward manner. Further, we provided an adaptive version of our algorithm whose convergence guarantee automatically improves when the underlying distribution has low variance. Finally, we provided an empirical evaluation showing that our algorithm matches the performance of the methods used to achieve state-of-the-art in training ResNet-50 on ImageNet and on BERT pretraining. Notably, these two tasks are often tackled with different optimizers (SGD and Adam respectively). Our method is one of the select few (You et al., 2019; Anil et al., 2019) which performs well on both simultaneously. Also, since NIGT does not maintain any second order statistic, it is significantly more memory efficient compared to popular optimizers such as Adam or LAMB.

Our results provide some intuition that can help explain the practical success of momentum in stochastic non-convex optimization. Prior analyses of momentum have failed to demonstrate significant theoretical benefits, but our results show that if one is willing to posit second-order smoothness, momentum may play a role in accelerating convergence. However, there are several open problems remaining. First, we suspect that the upper bound M imposed on F in our adaptive analysis is unnecessary, and also that the use of an extra gradient sample per iteration can be removed. Second, we note that in the noise-free case with second-order smoothness, the best-known dimension-free convergence rate is actually $O(1/\epsilon^{1.75})$ (Carmon et al., 2017) rather than the $O(1/\epsilon^2)$ achieved by gradient descent. Is it possible to design an adaptive algorithm that interpolates between this rate and $O(1/\epsilon^{3.5})$ in the noisy setting?

Finally, our implementation employs a few extra learning rate heuristics popular in practice, notably linear warm-up and polynomial decay (Devlin et al., 2019), as well as scaling the learning rate by the norm of the weights (You et al., 2017). We found these heuristics to be better-performing than our adaptive algorithm, despite their lack of theoretical motivation. Understanding the principles underlying the success of these heuristics poses an interesting orthogonal question to understanding momentum, and we hope that our work inspires future investigation into these areas.

References

Allen-Zhu, Z. Natasha 2: Faster non-convex optimization than sgd. In *Advances in neural information processing systems*, pp. 2675–2686, 2018.

Anil, R., Gupta, V., Koren, T., and Singer, Y. Memory efficient adaptive optimization. In *NeurIPS*, 2019.

Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Srebro, N.,

and Woodworth, B. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

Arnold, S. M., Manzagol, P.-A., Babanezhad, R., Mitliagkas, I., and Roux, N. L. Reducing the variance in online optimization by transporting past gradients. *arXiv preprint arXiv:1906.03532*, 2019.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Convex until proven guilty: Dimension-free acceleration of gradient descent on non-convex functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 654–663. JMLR. org, 2017.

Cutkosky, A. and Orabona, F. Momentum-based variance reduction in non-convex sgd. In *NeurIPS*, 2019.

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.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*, 2010.

Fang, C., Li, C. J., Lin, Z., and Zhang, T. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pp. 689–699, 2018.

Fang, C., Lin, Z., and Zhang, T. Sharp analysis for nonconvex sgd escaping from saddle points. In *Conference on Learning Theory*, pp. 1192–1234, 2019.

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

Goyal, P., Dollr, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2017.

Hazan, E., Levy, K., and Shalev-Shwartz, S. Beyond convexity: Stochastic quasi-convex optimization. In *Advances in Neural Information Processing Systems*, pp. 1594–1602, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Li, X. and Orabona, F. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 983–992, 2019.

Nesterov, Y. E. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Raffel, C., Shazeer, N., Roberts, A. K., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683, 2019.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*, 2018.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

Tran-Dinh, Q., Pham, N. H., Phan, D. T., and Nguyen, L. M. A hybrid stochastic optimization framework for stochastic composite nonconvex optimization. *arXiv preprint arXiv:1907.03793*, 2019.

Tripuraneni, N., Stern, M., Jin, C., Regier, J., and Jordan, M. I. Stochastic cubic regularization for fast nonconvex optimization. In *Advances in neural information processing systems*, pp. 2899–2908, 2018.

Ward, R., Wu, X., and Bottou, L. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pp. 6677–6686, 2019.

You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

You, Y., Li, J., Hseu, J., Song, X., Demmel, J., and Hsieh, C.-J. Reducing bert pre-training time from 3 days to 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Zaheer, M., Reddi, S., Sachan, D., Kale, S., and Kumar, S. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, pp. 9793–9803, 2018.

Zhou, D., Xu, P., and Gu, Q. Stochastic nested variance reduction for nonconvex optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 3925–3936. Curran Associates Inc., 2018.