Time-Delay Momentum: A Regularization Perspective on the Convergence and Generalization of Stochastic Momentum for Deep Learning

Zhang, Z.; Xu, W.; Sullivan, A.

TR2018-206 March 16, 2019

Abstract

In this paper we study the problem of convergence and generalization error bound of stochastic momentum for deep learning from the perspective of regularization.

arXiv

Time-Delay Momentum: A Regularization Perspective on the Convergence and Generalization of Stochastic Momentum for Deep Learning

Ziming Zhang[1] , **Wenju Xu**[2] , **Alan Sullivan**[1]
[1]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139-1955
[2]EECS, The University of Kansas1450 Jayhawk Blvd., Lawrence, KS 66045
{zzhang, sullivan}@merl.com, xuwenju@ku.edu

Abstract

In this paper we study the problem of convergence and generalization error bound of stochastic momentum for deep learning from the perspective of regularization. To do so, we first interpret momentum as solving an ℓ_2-regularized minimization problem to learn the offsets between arbitrary two successive model parameters. We call this *time-delay momentum* because the model parameter is updated after a few iterations towards finding the minimizer. We then propose our learning algorithm, *i.e.* stochastic gradient descent (SGD) with time-delay momentum. We show that our algorithm can be interpreted as solving a sequence of strongly convex optimization problems using SGD. We prove that under mild conditions our algorithm can converge to a stationary point with rate of $O(\frac{1}{\sqrt{K}})$ and generalization error bound of $O(\frac{1}{\sqrt{n\delta}})$ with probability at least $1-\delta$, where K, n are the numbers of model updates and training samples, respectively. We demonstrate the empirical superiority of our algorithm in deep learning in comparison with the state-of-the-art deep learning solvers.

1 Introduction

Deep learning has achieved big success in many applications such as object recognition in computer vision and speech translation in natural language processing. How to train such deep models well, however, has brought many challenges to the machine learning community, mainly because of huge dimensionality and high nonconvexity in such models.

Momentum [Sutskever *et al.*, 2013] is an important technique used in deep learning to accelerate the training. Its basic idea is to accumulate the history of weighted gradients to find the principal direction for updating network parameters. One popular formulation for momentum is as follows:

$$\triangle\omega_{t+1} = \mu_t \triangle\omega_t - \eta_t f'(\omega_t), \quad (1)$$
$$\omega_{t+1} = \omega_t + \triangle\omega_{t+1}, \quad (2)$$

where f' denotes the subgradient of function f (or its stochastic estimator), ω_t denotes the network parameter at the t-th iteration, $\triangle\omega_t$ denotes the offset between the two successive solutions, μ_t and η_t denote the corresponding momentum parameter and the learning rate, respectively.

Understanding the convergence and generalization error of deep learning with momentum, however, still remains elusive. Due to its convergence acceleration in convex optimization [Loizou and Richtárik, 2017], it is commonly believed that momentum conducts similar functionality in deep learning. Meanwhile many empirical works such as [Zhang *et al.*, 2018] have suggested that momentum is related to the generalization of learned models as well. Few theoretical works, however, are found to study both simultaneously.

Motivation: In this paper we interpret the role of momentum from the perspective of *regularization*. We are inspired by the functionality of weight decay for SGD as follows:

Claim (Weight Decay = ℓ_2 Regularization [Loshchilov and Hutter, 2019]). *Weight decay updates the weights ω based on*

$$\omega_{t+1} = (1-\lambda)\omega_t - \alpha f'_t(\omega_t) \quad (3)$$

at the t-th iteration, where λ, α denote the weight decay parameter and base learning rate, respectively. Then SGD with weight decay is equivalent to being executed without weight decay on $f_t^{reg}(\omega) = f_t(\omega) + \frac{\lambda'}{2}\|\omega\|_2^2$, with $\lambda' = \frac{\lambda}{\alpha}$.

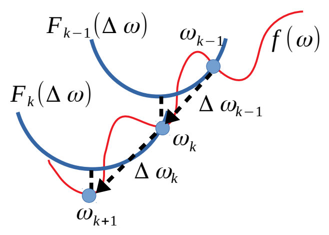


Figure 1: Illustration of momentum interpretation from regularization perspective.

By drawing an analogy between conventional momentum in Eq. 1 and weight decay in Eq. 3, we argue that momentum can be rethought from the regularization perspective as well. Different from weight decay that is regularized on the network parameter ω directly to prevent it from being too far away from the origin, our novel momentum interpretation regularizes the offset $\triangle\omega$ from the current parameter. In this way, gradient descent (GD) with momentum in Eq. 1 and Eq. 2 can be interpreted as an attempt to solve a certain regularized optimization problem to learn the offset *conditioned* on the current solution. Fig. 1 illustrates our interpretation above, where F's are the (auxiliary) regularized functions.

Contributions: We summarize the main contributions of our paper as follows:

C1. We propose a novel time-delay momentum based on the regularization interpretation. Our learning algorithm, *i.e.* SGD with time-delay momentum, provides simpler yet deeper insight on the convergence and generalization of stochastic momentum.

C2. We show that our algorithm can be interpreted as a sequential convex optimization algorithm where SGD is used to solve a sequence of strongly convex problems for determining the offsets.

C3. We prove the convergence rate and generalization error bound of our learning algorithm.

C4. We demonstrate that empirically our algorithm works on par with or even better than the state-of-the-art deep learning solvers such as Adam [Kingma and Ba, 2014] and RMSProp [Hinton *et al.*, 2012], with the same computational complexity as SGD. .

1.1 Related Work

Convergence: Most of related work on momentum lies in deterministic optimization [Bottou *et al.*, 2018]. For instance, in [Nesterov, 1998] accelerated gradient descent was introduced for smooth convex optimization, and has been found to help escape saddle points faster than GD in deep learning [Jin *et al.*, 2017]. The convergence analysis has been applied to smooth nonconvex optimization as well [Ochs *et al.*, 2014; Ochs, 2018]. In [Ghadimi and Lan, 2016] a stochastic accelerated gradient method was proposed for nonconvex optimization with convergence analysis that is hardly used to explain the scenarios with momentum. Recently it has been proved in [Loizou and Richtárik, 2017] that SGD with momentum can achieve global nonassymptotic linear convergence rate in convex optimization. In [Reddi *et al.*, 2018] the convergence analysis was provided for Adam which involves momentum in learning, similarly in [Basu *et al.*, 2018] for Adam and RMSProp, and in [Li and Orabona, 2018] for a modification of Adagrad [Duchi *et al.*, 2011].

In [Shalev-Shwartz *et al.*, 2009] it revealed that although a stochastic convex optimization problem with batches is learnable, there is no uniform convergence in the general case, and thus empirical minimization might fail, but strongly convex problems are learnable using empirical minimization. The key insight here comes from regularization.

In [Dinh and Diehl, 2010] a local convergence analysis was provided for sequential convex programming (optimization) based on the concept of local contraction, which is widely used in numerical analysis for fixed-point solutions [Agarwal *et al.*, 2001]. More generally the analysis can be extended using inexact Newton methods [Dembo *et al.*, 1982].

Generalization Error: Most of related work focuses on the generalization error of SGD, rather than SGD with momentum, for convex and nonconvex optimization. For instance, in [Shalev-Shwartz *et al.*, 2009] it proved that for strongly convex functions with probability of at least $1-\delta$ stochastic convex optimization can achieve the error bound of $O(\frac{1}{\sqrt{\delta n}})$, where n is the number of training samples.

For nonconvex optimization, the stability [Bousquet and Elisseeff, 2002] and PAC Bayesian theory [McAllester, 1999] are often used to analyze SGD. For instance, [Hardt *et al.*, 2015] first applied the stability framework to study the expected error, and [Kuzborskij and Lampert, 2017] further provided a data-dependent generalization error bound. [Mou *et al.*, 2017] applied the PAC Bayesian theory to study the error with additive Gaussian noise, while [London, 2017] provided the error bound with probabilistic guarantee for strongly convex loss functions by combining the stability framework with the PAC Bayesian theory. [Zhou *et al.*, 2018] established various generalization error bounds with probabilistic guarantee. Especially with strongly convex regularizers SGD can achieve the error bound of $O(\frac{1}{\sqrt{\delta n}})$ as well.

Remarks: We notice that [Yan *et al.*, 2018] has proposed a unified analysis on stochastic momentum for deep learning. The proposed SUM algorithm, an interpolation of several stochastic momentum methods, can achieve the convergence rate of $O(\frac{1}{\sqrt{T}})$ where T denotes the number of iterations in SGD. The generalization error bound, however, is not clear in their analysis.

In contrast, in this paper we study the problem of convergence and generalization from the perspective of regularization. We prove that our algorithm can converge with the same rate as [Yan *et al.*, 2018]. More importantly, we also prove the generalization error bound. Our algorithm can be interpreted as solving a sequence of strongly convex optimization problems, and thus our theoretical analysis is more straightforward and easier to understand.

2 Time-Delay Momentum

2.1 Problem Definition

Let us consider the following optimization problem:

$$\min_{\omega} f(\omega) = \mathbb{E}_{x\sim\mathcal{X}}[\ell(\omega; x)], \tag{4}$$

where $\omega \in \mathbb{R}^d$ denotes the model parameters (*e.g.* network weights), $x \sim \mathcal{X}$ denotes a data sample drawn from a distribution $\mathcal{X}$, $\ell : \mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$ denotes a loss function, and $\mathbb{E}_{x\sim\mathcal{X}}$ denotes the expectation operator over x.

Assumption. *For all $x \sim \mathcal{X}$ the loss function satisfies:*

A1. $\ell(\cdot; x)$ is (possibly nonconvex) nonnegative, Lipschitz-continuously differentiable, and L-smooth (so does f);

A2. $|\ell(\cdot; x)|$ is uniformly bounded (so does f);

A3. For any fixed data set $\mathcal{S}$ and any ξ that is generated uniformly from $\{1, \cdots, n\}$ at random, there exists a constant $\nu_{\mathcal{S}} > 0$ such that for all $\omega \in \Omega$ one has

$$\mathbb{E}_\xi \left\| \ell'(\omega; x_\xi) - \frac{1}{n}\sum_{k=1}^{n} \ell'(\omega; x_k) \right\|_2^2 \le \nu_{\mathcal{S}}^2, \tag{5}$$

where ℓ' denotes the subgradient of ℓ, $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector;

A4. For any fixed data set $\mathcal{S}$ and any ξ that is generated uniformly from $\{1, \cdots, n\}$ at random, there exists a constant $G_{\mathcal{S}} > 0$ such that for all $\omega \in \Omega$ one has

$$\mathbb{E}_\xi \left\| \ell'(\omega; x_\xi) \right\|_2^2 \le G_{\mathcal{S}}^2. \tag{6}$$

Algorithm 1 SGD with Time-Delay Momentum

```
Input : loss $\ell$, learning rate $\{\eta_{k,t}\}$, momentum $\{\mu_{k,t}\}$, training data $\mathcal{S}$, parameters $K, T$
Output: minimizer $\omega$
Randomly initialize $\omega \leftarrow \omega_1$;
for $k = 1, 2, \cdots, K-1$ do
    $\triangle\omega_{k,1} \leftarrow \mathbf{0}$;
    for $t = 1, 2, \cdots, T-1$ do
        Uniformly draw a training sample $x_t \in \mathcal{S}$ at random;
        $\triangle\omega_{k,t+1} \leftarrow \mu_{k,t}\triangle\omega_{k,t} - \eta_{k,t}\ell'(\omega_k + \triangle\omega_{k,t}; x_t)$;
    end
    $\omega_{k+1} \leftarrow \omega_k + \triangle\omega_{k,T}$;
end
return $\omega \leftarrow \omega_K$;
```

2.2 Learning Algorithm

Inspired by regularization interpretation of weight decay, in this paper we propose the following update rule as regularization interpretation of momentum:

$$\min_{\triangle\omega} F_k(\triangle\omega) = f(\omega_k + \triangle\omega) + \frac{\theta_k}{2}\|\triangle\omega\|_2^2, \qquad (7)$$

where ω_k denotes the model parameter at the k-th update, $\triangle\omega$ denotes the offset between two successive solutions, and $\theta_k \geq 0$ is a predefined constant.

Eq. 7 aims to search for the minimizer $\triangle\omega^*$ to generate a new solution $\omega_{k+1} = \omega_k + \triangle\omega^*$ for the $(k+1)$-th update. Moreover, it has the following nice properties:

Proposition 1 (Monotonically Decreasing)**.** *Based on Eq. 7, we have*

$$f(\omega_k + \triangle\omega^*) \leq F_k(\triangle\omega^*) \leq F_k(\mathbf{0}) = f(\omega_k), \qquad (8)$$

where $\mathbf{0}$ *denotes a vector of zeros, and the equality holds if and only if* $\triangle\omega^* = \mathbf{0}$.

Proposition 2 (Stationary Points)**.** *The minimizer* $\triangle\omega^* = \mathbf{0}$ *is equivalent to that* $f'(\omega_k) = \mathbf{0}$ *holds, where* f' *denotes the subgradient of* f.

Proof. At $\triangle\omega^*$, we have $F_k'(\triangle\omega^*) = f'(\omega_k + \triangle\omega^*) + \theta_k\triangle\omega^* = \mathbf{0}$, where F_k' denotes the subgradient of F_k over $\triangle\omega$. If $\triangle\omega^* = \mathbf{0}$, then $f'(\omega_k) = \mathbf{0}$; vice versa. □

Proposition 3 (Upper-Bound)**.** *Let* f^* *be the global minimum of* f. *We have*

$$\|\triangle\omega^*\|_2 \leq \left(\frac{2}{\theta_k}\Big[f(\omega_k) - f^*\Big]\right)^{\frac{1}{2}}. \qquad (9)$$

Proof. Based on Prop. 1 above, we have $f^* + \frac{\theta_k}{2}\|\triangle\omega^*\|_2^2 \leq f(\omega_k + \triangle\omega^*) + \frac{\theta_k}{2}\|\triangle\omega^*\|_2^2 = F_k(\triangle\omega^*) \leq f(\omega_k)$. Then using simple algebra we can complete our proof. □

To solve Eq. 7 we propose using SGD and list our learning algorithm in Alg. 1 where $\mu_{k,t} = 1 - \theta_k\eta_{k,t}$. Compared with Eq. 1 we can easily see that the gradient of Eq. 7 can be considered as momentum, except that in our algorithm the model parameter is updated after $T-1$ iterations, rather than immediately after computing momentum as Eq. 2. Therefore, we name it *Time-Delay Momentum*, in contrast to the conventional one in Eq. 1.

Note that we refer to the outer-loop in Alg. 1 as *updates*, and the inner-loop as *iterations*. In this way our algorithm is equivalent to solving a sequence of $K-1$ optimization problems defined in Eq. 7. As we see later, the initialization of $\triangle\omega$ has effect on the generalization error bound, and any vector that guarantees $f(\omega_k + \triangle\omega) \leq f(\omega_k)$ can be used as initialization for better bounds. Also if we initialize $\triangle\omega$ using Eq. 1 as well as setting $T = 1$, we recover SGD with traditional momentum.

3 Theoretical Analysis

3.1 Preliminaries

Here we list some definitions and lemmas that are used in our analysis later. We follow the notations in the reference papers with definitions so that they are self-explainable.

Definition 1 (Lipschitz Continuity [Nesterov, 1998])**.** *Function* f *is* Lipschitz continuous *on* $\mathcal{X}$, *if for some constant* $L \geq 0$ *we have*

$$|f(x) - f(y)| \leq L|x - y|, \forall x, y \in \mathcal{X}. \qquad (10)$$

Definition 2 (λ-Strongly Convex [Nesterov, 1998])**.** *A continuously differentiable function* f *is called strongly convex on a convex set* $\mathcal{X}$ *if for some constant* $\lambda > 0$ *we have*

$$f(y) \geq f(x) + \langle f'(x), y - x\rangle + \frac{\lambda}{2}\|y - x\|_2^2, \forall x, y \in \mathcal{X}. \qquad (11)$$

Lemma 1 ([Nesterov, 1998])**.** *Given an* L*-smooth function* f, *i.e. its derivatives are Lipschitz continuous with constant* L, *we have* $\forall x, y$,

$$f(y) \geq f(x) + \langle f'(x), y - x\rangle + \frac{1}{2L}\|f'(y) - f'(x)\|_2^2, \qquad (12)$$

where $\langle\cdot,\cdot\rangle$ *denotes the inner product operator.*

Lemma 2 ([Rakhlin *et al.*, 2011])**.** *Suppose* F *is* λ*-strongly convex and* μ*-smooth with respect to* ω^* *over a convex set* Ω, *and that the stochastic subgradient of* F, $\hat{g}_t$, *satisfies* $\mathbb{E}[\|\hat{g}_t\|^2] \leq G^2$. *Then if we pick learning rate* $\eta_t = \frac{1}{\lambda t}$ *in SGD, it holds for any* T *that*

$$\mathbb{E}[F(\omega_T) - F(\omega^*)] \leq \frac{2\mu G^2}{\lambda^2 T}. \qquad (13)$$

Lemma 3 ([Zhou *et al.*, 2018])**.** *Consider the following two optimization problems:*

$$\min_{\omega\in\Omega} \Phi(\omega) = f(\omega) + \theta h(\omega), \qquad (14)$$

$$\min_{\omega\in\Omega} \Phi_{\mathcal{S}}(\omega) = f_{\mathcal{S}}(\omega) + \theta h(\omega), \qquad (15)$$

where h *denotes the* 1*-strongly convex and nonnegative regularizer and* $f, f_{\mathcal{S}}$ *are the population and empirical risks, respectively.*

Let Assmp. (A1-A3) hold for both f *and* $f_{\mathcal{S}}$, *and apply the proximal SGD* $\omega_{t+1} = prox_{\alpha_t,h}(\omega_t - \alpha_t\ell'(\omega_t; x_{\xi_t}))$ *to solve Eq. 15 with the data set* $\mathcal{S}$. *Let* $\theta > L$ *and* $\alpha_t = \frac{c}{t+2}$ *with*

$0 < c < \frac{1}{L}$*. Then, the following bound holds with probability at least* $1 - \delta$*:*

$$|\Phi(\omega_{T,\mathcal{S}}) - \Phi_{\mathcal{S}}(\omega_{T,\mathcal{S}})| \leq \sqrt{\frac{1}{n\delta}\left(C_1 + C_2\sqrt{L\Phi(\omega_0) + C3}\right)}, \tag{16}$$

where C_1, C_2, C_3 *are some constants (see [Zhou* et al.*, 2018] for more details),* n *is the number of training samples,* ω_0 *is the initialization, and* $\omega_{T,\mathcal{S}}$ *is the solution at the* t*-th iteration on the data set.*

3.2 Convergence

In this section we analysis the convergence of Alg. 1. We start with proving the smoothness and strong convexity of F_k in Eq. 7. Then based on the convergence of SGD in Lemma 2 we derive the upper bound for the expectation of $\|\triangle\omega_{k,T}\|$.

Lemma 4. F_k *is a* $(\theta_k + L)$*-smooth function.*

Proof. Based on Def. 1, for any $\triangle\omega_1, \triangle\omega_2$, we have

$$\begin{aligned}&\|F'_k(\triangle\omega_1) - F'_k(\triangle\omega_2)\|_2\\ =&\|f'(\omega_k + \triangle\omega_1) - f'(\omega_k + \triangle\omega_2) + \theta_k(\triangle\omega_1 - \triangle\omega_2)\|_2\\ \leq&\|f'(\omega_k + \triangle\omega_1) - f'(\omega_k + \triangle\omega_2)\|_2 + \theta_k\|\triangle\omega_1 - \triangle\omega_2\|_2\\ \leq&(\theta_k + L)\|\triangle\omega_1 - \triangle\omega_2\|_2.\end{aligned} \tag{17}$$

Based on Def. 1 we can complete our proof. □

Lemma 5 (Strong Convexity of F_k)**.** *If* $\theta_k > L \geq 0$ *holds,* $F_k(\triangle\omega)$ *in Eq. 7 is* λ_k*-strongly convex with* $\lambda_k = \frac{(\theta_k - L)^2}{\theta_k + L}$*.*

Proof. Based on Eq. 17 and the triangle inequality we have

$$\begin{aligned}&\|F'_k(\triangle\omega_1) - F'_k(\triangle\omega_2)\|_2\\ &\quad\geq |d_1 - d_2| \geq (\theta_k - L)\|\triangle\omega_1 - \triangle\omega_2\|_2,\end{aligned} \tag{18}$$

where we define $d_1 = \|f'(\omega_k + \triangle\omega_1) - f'(\omega_k + \triangle\omega_2)\|_2 \leq L\|\triangle\omega_1 - \triangle\omega_2\|_2$ and $d_2 = \theta_k\|\triangle\omega_1 - \triangle\omega_2\|_2$. Further based on Lemma 4 and Lemma 1 we have

$$\begin{aligned}F_k(\triangle\omega_1) \geq& F_k(\triangle\omega_2) + \langle F'_k(\triangle\omega_2), \triangle\omega_1 - \triangle\omega_2\rangle\\ &+ \frac{(\theta_k - L)^2}{2(\theta_k + L)}\|\triangle\omega_1 - \triangle\omega_2\|_2^2.\end{aligned} \tag{19}$$

Based on Def. 2 we then can complete our proof. □

Corollary 1 (Unique Existence)**.** *Given* λ_k *and* L*, there always exists a unique* θ_k *in Eq. 7 so that Lemma 5 holds.*

Proof. Letting $h(\theta_k) = \theta_k^2 - (2L + \lambda_k)\theta_k + (L^2 - \lambda_k L)$, then $h(\theta_k) = 0$ leads to λ_k in Lemma 5. Because of $h(L) = -2\lambda_k L \leq 0$ and the properties of quadratic function, we can complete our proof. □

Theorem 1 (Convergence of Alg. 1)**.** *Let Assmp. A1 and A4 hold, and* $\eta_{k,t} = \frac{1}{\lambda_k t}, \lambda_k = \frac{(\theta_k - L)^2}{\theta_k + L}, \theta_k > L, \forall k, \forall t$*. Then Alg. 1 guarantees to converge to a stationary point of* f *with rate of* $O(\frac{1}{\sqrt{K}})$*, in expectation, when* T *is sufficiently large as a constant.*

Proof. Based on the assumptions in Thm. 1, $F_k, \forall k$ in Eq. 7 is a smooth and strongly convex function. Therefore, given the learning rate and Lemma 2, with sufficiently large T we have $\mathbb{E}_{\triangle\omega_{k,T}|\omega_k}[F_k(\triangle\omega_{k,T}) - F_k(\triangle\omega_k^*)] \approx 0$ where $\triangle\omega_k^*$ denotes the minimizer of Eq. 7 conditioned on ω_k. This implies that $\mathbb{E}_{\triangle\omega_{k,T}|\omega_k}[F_k(\triangle\omega_{k,T}) - F_k(\mathbf{0})] \leq 0$ holds based on Prop. 1. Therefore, we have

$$\begin{aligned}&f(\omega_1) - f^* \geq f(\omega_1) - \lim_{K\to+\infty}\mathbb{E}_{\omega_K|\omega_1,\cdots,\omega_{K-1}}[f(\omega_K)]\\ &= \sum_{k=1}^{+\infty}\mathbb{E}_{\omega_k|\omega_1,\cdots,\omega_{k-1}}\left[f(\omega_k) - \mathbb{E}_{\omega_{k+1}|\omega_k}[f(\omega_{k+1})]\right]\\ &\geq \sum_{k=1}^{+\infty}\frac{\theta_k}{2}\mathbb{E}_{\triangle\omega_{k,T}}\left[\|\triangle\omega_{k,T}\|_2^2\right], \theta_k > L \geq 0, \forall k.\end{aligned} \tag{20}$$

Besides, based on Prop. 3 we know that $\triangle\omega_{k,T}, \forall k$ is upper-bounded. Then we conclude $\mathbb{E}_{\triangle\omega_{k,T}}\left[\|\triangle\omega_{k,T}\|_2^2\right] = O\left(\frac{1}{k^p}\right)$ with $\exists p > 1$ (otherwise Eq. 20 cannot hold). Since it holds that $\left(\mathbb{E}_{\triangle\omega_{k,T}}\left[\|\triangle\omega_{k,T}\|_2\right]\right)^2 \leq \mathbb{E}_{\triangle\omega_{k,T}}\left[\|\triangle\omega_{k,T}\|_2^2\right]$, we further have

$$\mathbb{E}_{\triangle\omega_{k,T}}\left[\|\triangle\omega_{k,T}\|_2\right] = O\left(\frac{1}{k^{\frac{p}{2}}}\right) < O\left(\frac{1}{\sqrt{k}}\right). \tag{21}$$

leading to $\lim_{k\to+\infty}\mathbb{E}_{\triangle\omega_{k,T}}\left[\|\triangle\omega_{k,T}\|_2\right] = 0$. Based on Prop. 2 we can complete our proof. □

Discussion: By only taking the times of parameter updates into account, our convergence rate has the same order of that in [Yan *et al.*, 2018]. Differently, our convergence analysis is based on strong convexity of the auxiliary function F_k in Eq. 7 and convergent series, leading to much simpler analysis.

3.3 Generalization Error Bound

In this section, we derive our generalization bound based on Lemma 3. We show that under mild conditions our algorithm can satisfy the assumptions in Lemma 2 and Lemma 3 simultaneously to validate both convergence and generalization.

Theorem 2 (Generalization Error Bound of Alg. 1)**.** *Let Assmp. (A1-A4) hold, and* $\eta_{k,t} = \frac{1}{\lambda_k t}, \lambda_k = \frac{(\theta_k - L)^2}{\theta_k + L}, \theta_k > \left(\frac{5+\sqrt{33}}{2}\right)L \approx 5.37L, \forall k, \forall t$*. Then the following bound holds with probability at least* $1 - \delta$*:*

$$\begin{aligned}&|f(\omega_{k+1}) - f_{\mathcal{S}}(\omega_{k+1})|\\ &\leq \sqrt{\frac{1}{n\delta}\left(C_1 + C_2\sqrt{Lf(\omega_k) + C3}\right)} \approx O\left(\frac{f(\omega_k)^{1/4}}{(n\delta)^{1/2}}\right),\end{aligned} \tag{22}$$

where $f_{\mathcal{S}}$ *denotes the empirical risk of* f *on the data set* $\mathcal{S}$*, i.e.* $f_{\mathcal{S}}(\omega_k) = \frac{1}{|\mathcal{X}|}\sum_{x_t\in\mathcal{X}}\ell(\omega; x_t)$*,* $\mathcal{X}$ *denotes the training set, and* $|\mathcal{X}| = n$ *denotes the number of training samples.*

Proof. In order to satisfy the learning rate conditions in Lemma 2 and Lemma 3 (recall that η_t and α_t are the learning rates accordingly), we set $c = \frac{t+2}{\lambda_k t}$ in Lemma 3 so that $\eta_{k,t} = \eta_t = \alpha_t = \frac{1}{\lambda_k t}$. Then we need to show that it holds that $c = \frac{t+2}{\lambda_k t} < \frac{1}{L}$ with $\lambda_k = \frac{(\theta_k - L)^2}{\theta_k + L}, \theta_k > \left(\frac{5+\sqrt{33}}{2}\right)L, \forall k, \forall t.$

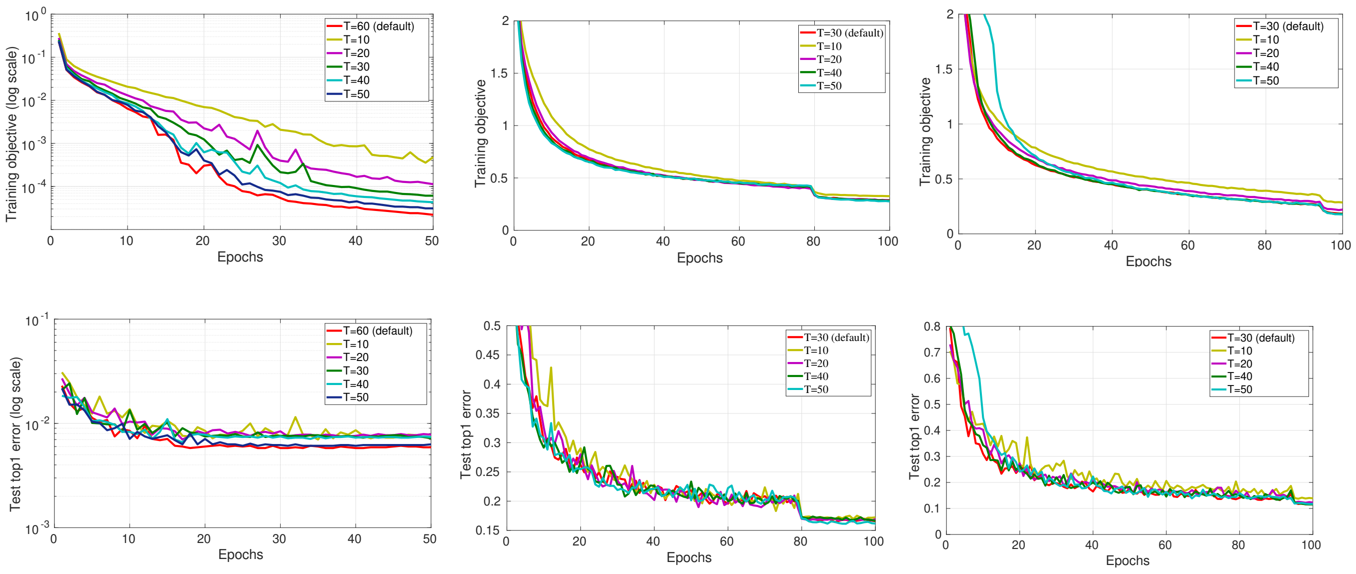

(a) LeNet-5 on MNIST (b) LeNet on CIFAR-10 (c) NIN on CIFAR-10

Figure 2: Illustration of training and testing behavior with different T's in our algorithm.

By solving $\frac{t+2}{\lambda_k t} < \frac{1}{L}$ we derive $\theta_k > \frac{3t+2+\sqrt{9t^2+20t+4}}{2t}L$ where the RHS is monotonically decreasing *w.r.t.* $t \geq 1$, and thus upper bounded by $\left(\frac{5+\sqrt{33}}{2}\right)L$ when $t = 1$. Therefore, given the conditions in Thm. 2 above, the learning rate conditions hold for both convergence and generalization.

By fitting Alg. 1 into Lemma 3, we have $h(\cdot) = \frac{1}{2}\|\cdot\|_2^2$, and accordingly the proximal SGD in Lemma 3 is equivalent to SGD used in Alg. 1. By redefining $\Phi(\omega) \equiv F_k(\triangle\omega), \Phi_{\mathcal{S}}(\omega) \equiv F_{k,\mathcal{S}}(\triangle\omega), \omega_0 \equiv \mathbf{0}$, and substituting these to Eq. 16, we can achieve Eq. 22. If $f(\omega_k)$ dominates the RHS, we then have the big O approximation.

By doing so, we now can complete our proof. $\square$

Discussion: Our error bound in Eq. 22 shows that the generalization error of deep learning depends on not only the size of training sample (as well as the probability level) but also the *population risk of the initialization*. This result may provide some insight on why pretrained networks can work well on new data sets, even without fine-tuning. From the power factors it seems that number of training samples is more important than the initialization for the success of deep learning.

It is worth mentioning that our algorithm may achieve better generalization error bounds that has *exponential* decay in probability as $n \to \infty$ and $T \to \infty$ based on Thm. 6 in [Zhou *et al.*, 2018]. However, this requires much stronger condition on c, *i.e.* $\frac{1}{2(\lambda_k - L)} < c < \frac{1}{\lambda_k - L}$. We will consider exploring more on improving our generalization error bound in our future work.

4 Experiments

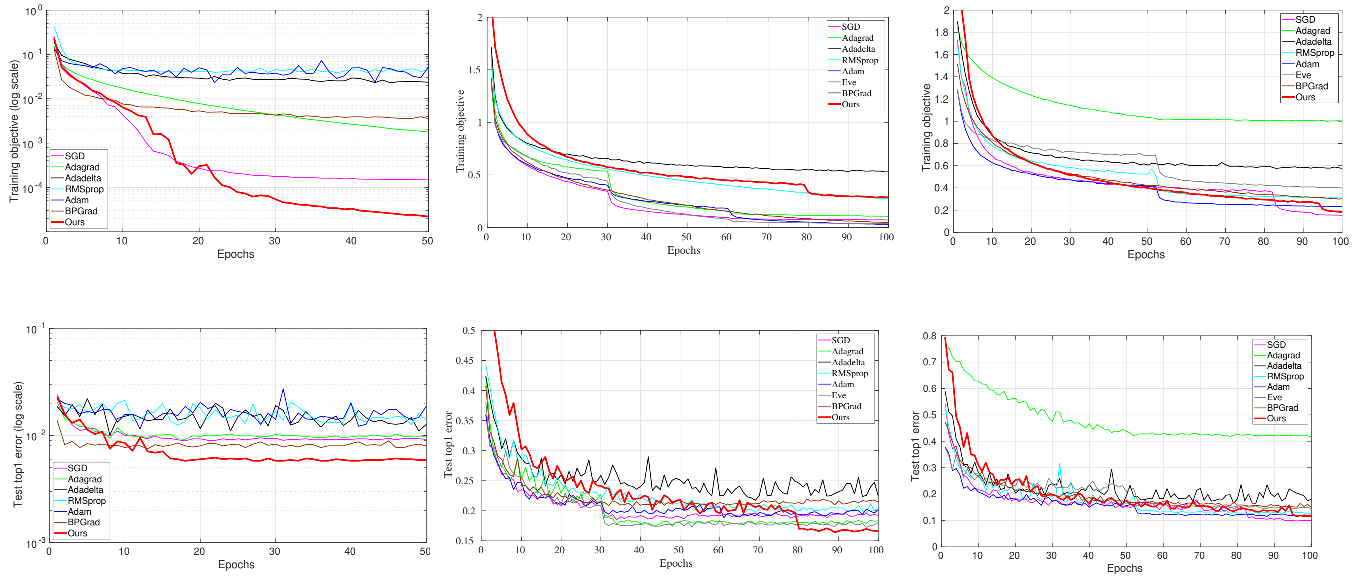
To demonstrate the convergence and generalization of our algorithm, we utilize it to train deep models for image classification and compare our performance with state-of-the-art deep learning solvers, including SGD with momentum, Adagrad, Adadelta [Zeiler, 2012], RMSProp, Adam, Eve [Koushik and Hayashi, 2016], and BPGrad [Zhang *et al.*, 2018]. For fair comparison, we tune the hyper-parameters of these solvers to generate the best classification accuracy.

Data Sets: We test our algorithm on the MNIST [LeCun, 1998] and CIFAR-10 [Krizhevsky, 2009] data sets. MNIST consists of 60,000 training images and 10,000 testing images in 10 classes with resolution of 28×28, while CIFAR-10 consists of 10 object classes of natural images with 50,000 training images and 10,000 test images with resolution of 32×32.

Networks: We employ three different networks for demonstration. LeNet [Lecun *et al.*, 1998] and its variant, LeNet-5, consist of a few convolutional, max-pool, and multilayer perceptron (MLP) layers. Network in network (NIN) [Lin *et al.*, 2014] introduces a mlpconv layer in convolutional neural networks (CNNs) which maps the input local patch to the output feature vector with an MLP consisting of multiple fully connected layers with nonlinear activation functions. For all the three networks, we use the same architectures as the ones from https://github.com/vlfeat/matconvnet/tree/master/examples/cifar.

Training Protocols: For each of the baseline solvers on MNIST, we train the networks for 50 epochs with a mini-batch size 100, weight decay 0.0005, and momentum 0.9. The global learning rate is set to 0.001 for Adagrad, RMSProp, Adam and SGD. Adadelta does not require the global learning rate. Similarly, on CIFAR-10 we train an individual model for 100 epochs with a mini-batch size 100, weight decay 0.0005, and momentum 0.9. The global learning rate is set to 0.001 for RMSProp, but 0.01 for Adagrad, Adam and Eve, and it is reduced to 0.005 and 0.001 at the 31st and 61st epochs. The initial learning rate for SGD is 0.05 , and it is multiplied by 0.1 at the 31st and 61st epochs.

For our approach on both data sets, we train the networks



(a) LeNet-5 on MNIST (b) LeNet on CIFAR-10 (c) NIN on CIFAR-10

Figure 3: Illustration of training and testing behavior of different deep learning solvers on different data sets using different networks.

with a mini-batch size 100, momentum 0.9, and global learning rate 0.01. We use the learning rate decay of factor 0.1 to accelerate the convergence on CIFAR-10 as other baselines. We tried the adaptive learning rate as in Thm. 2, but observed that the inner loop requires much more iterations to converge to a stable solution, leading to inferior performance within the same number of epochs than the baselines. We therefore use the fixed learning rate with decay as described above for fast convergence, which is widely adopted in deep learning. Note that with more epochs our approach with adaptive learning rate can achieve very comparable or even better performance than that with fixed learning rate.

Results: In our algorithm the parameter T controls the precision of the solution to the optimality of Eq. 7, roughly speaking, as well as the number of model updates per epoch. For instance, on MNIST if setting $T = 60$, we have $60,000/(100 \times T) = 10$ model updates per epoch. To validate the effect of T on empirical performance, we first show some comparison of different T's in Fig. 2. In general as we see, all the curves in each subfigure share the same convergence behavior in both training and testing, where the performance gaps are marginal. From Fig. 2(c) it seems that larger T does not bring much benefit in terms of convergence and generalization, although it potentially can reach better models towards the optimality, but smaller T tends to achieve slightly worse performance. Another interesting observation is that the performance of our algorithm is quite insensitive to the number of model updates. This will be very useful in distributed deep learning [Dean *et al.*, 2012] because it may reduce the communication cost between local machines. This topic is out of scope of this paper, and we will consider it in our future work.

Further we compare our algorithm with state-of-the-art deep learning solvers in Fig. 3, where the curves of our approach are based on the default T's in Fig. 2. As we see, our approach works on par with or even better than these baselines. Compared with SGD, our approach start with slower convergence, but gradually catch up with it later. The convergence curves of SGD and ours are quite similar, indicating that the two methods may share the same convergence rate as our analysis in Thm. 1. Especially in Fig. 3(b) our approach achieves relatively larger training loss than most of the baselines, but the best testing error. This observation suggests that the generalization of deep models may not be related to the empirical risk directly, although it can be used to estimate the population risk. In summary, by searching for models that are close to the optimality in Eq. 7, our algorithm can generalize well in training different networks on different data sets.

5 Conclusion

In this paper we propose a novel time-delay momentum for analyzing the convergence and generalization error bound of stochastic momentum in deep learning from the perspective of regularization. We propose a regularized formulation to learn the parameter offsets for updating the models. We show that our learning algorithm based on the regularized formulation can be interpreted as a sequential convex optimization algorithm, where SGD is used to solve a sequence of strongly convex problems. We further prove that under mild conditions our algorithm can converge to a stationary point with the rate of $O(\frac{1}{\sqrt{K}})$ and the generalization error bound of $O(\frac{1}{\sqrt{n\delta}})$. Our convergence result is consistent with recent results in [Yan *et al.*, 2018], while our generalization result is novel. We believe that such results may provide some simpler yet deeper insight for understanding stochastic momentum in deep learning. We demonstrate the empirical superiority of our algorithm on image classification with comparison with state-of-the-art deep learning solvers.

References

[Agarwal *et al.*, 2001] Ravi P. Agarwal, Maria Meehan, and Donal O'Regan. page 159–168. Cambridge Tracts in Mathematics. Cambridge University Press, 2001.

[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.

[Bottou *et al.*, 2018] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[Bousquet and Elisseeff, 2002] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[Dean *et al.*, 2012] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *NIPS*, pages 1223–1231, 2012.

[Dembo *et al.*, 1982] Ron S Dembo, Stanley C Eisenstat, and Trond Steihaug. Inexact newton methods. *SIAM Journal on Numerical analysis*, 19(2):400–408, 1982.

[Dinh and Diehl, 2010] Quoc Tran Dinh and Moritz Diehl. Local convergence of sequential convex programming for nonconvex optimization. In *Recent Advances in Optimization and its Applications in Engineering*, pages 93–102. Springer, 2010.

[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.

[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.

[Hardt *et al.*, 2015] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

[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.

[Jin *et al.*, 2017] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. *arXiv preprint arXiv:1711.10456*, 2017.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Koushik and Hayashi, 2016] Jayanth Koushik and Hiroaki Hayashi. Improving stochastic gradient descent with feedback. *arXiv preprint arXiv:1611.01505*, 2016.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[Kuzborskij and Lampert, 2017] Ilja Kuzborskij and Christoph H Lampert. Data-dependent stability of stochastic gradient descent. *arXiv preprint arXiv:1703.01678*, 2017.

[Lecun *et al.*, 1998] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LeCun, 1998] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[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.

[Lin *et al.*, 2014] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *International Conference on Learning Representations*, 2014.

[Loizou and Richtárik, 2017] Nicolas Loizou and Peter Richtárik. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *arXiv preprint arXiv:1712.09677*, 2017.

[London, 2017] Ben London. A pac-bayesian analysis of randomized learning with application to stochastic gradient descent. In *NIPS*, pages 2931–2940, 2017.

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

[McAllester, 1999] David A McAllester. Pac-bayesian model averaging. In *Proc. 12th Annual Conference on Computational Learn-ing Theory*, 1999.

[Mou *et al.*, 2017] Wenlong Mou, Liwei Wang, Xiyu Zhai, and Kai Zheng. Generalization bounds of sgld for non-convex learning: Two theoretical viewpoints. *arXiv preprint arXiv:1707.05947*, 2017.

[Nesterov, 1998] Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course. 1998.

[Ochs *et al.*, 2014] Peter Ochs, Yunjin Chen, Thomas Brox, and Thomas Pock. ipiano: Inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences*, 7(2):1388–1419, 2014.

[Ochs, 2018] Peter Ochs. Local convergence of the heavy-ball method and ipiano for non-convex optimization. *Journal of Optimization Theory and Applications*, pages 1–28, 2018.

[Rakhlin *et al.*, 2011] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

[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.

[Shalev-Shwartz *et al.*, 2009] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *COLT*, 2009.

[Sutskever *et al.*, 2013] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[Yan *et al.*, 2018] Yan Yan, Tianbao Yang, Zhe Li, Qihang Lin, and Yi Yang. A unified analysis of stochastic momentum methods for deep learning. In *IJCAI*, 2018.

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

[Zhang *et al.*, 2018] Ziming Zhang, Yuanwei Wu, and Guanghui Wang. Bpgrad: Towards global optimality in deep learning via branch and pruning. In *CVPR*, pages 3301–3309, 2018.

[Zhou *et al.*, 2018] Yi Zhou, Yingbin Liang, and Huishuai Zhang. Generalization error bounds with probabilistic guarantee for sgd in nonconvex optimization. *arXiv preprint arXiv:1802.06903*, 2018.