Local Regularizer Improves Generalization

Yikai Zhang,[1*] Hui Qu,[1*] Dimitris Metaxas,[1] Chao Chen[2]
[1]Department of Computer Science, Rutgers University
[2]Departments of Biomedical Informatics, Stony Brook University
{yz422, hui.qu, dnm}@cs.rutgers.edu, chao.chen.cchen@gmail.com

Abstract

Regularization plays an important role in generalization of deep learning. In this paper, we study the generalization power of an unbiased regularizor for training algorithms in deep learning. We focus on training methods called Locally Regularized Stochastic Gradient Descent (LRSGD). An LRSGD leverages a proximal type penalty in gradient descent steps to regularize SGD in training. We show that by carefully choosing relevant parameters, LRSGD generalizes better than SGD. Our thorough theoretical analysis is supported by experimental evidence. It advances our theoretical understanding of deep learning and provides new perspectives on designing training algorithms. The code is available at https://github.com/huiqu18/LRSGD.

1 Introduction

Deep learning has achieved great success in practice. Due to its expressive power, deep models can easily fit data and achieve sufficiently low empirical risk (Raghu et al. 2017; Lin and Jegelka 2018; Cohen, Sharir, and Shashua 2016; Brutzkus et al. 2017). Meanwhile, it is crucial to have a better control of the generalization performance. In recent years, substantial work has been done toward understanding and improving generalization of deep nets. (Zhang et al. 2016; Belkin, Ma, and Mandal 2018; Belkin, Hsu, and Mitra 2018; Neyshabur et al. 2017; Arora et al. 2018).

In machine learning, regularization is known to improve generalization both theoretically and empirically (Tibshirani 1996; Hoerl and Kennard 2000; Krogh and Hertz 1992; Bousquet and Elisseeff 2002; Friedman, Hastie, and Tibshirani 2001). In addition to classical regularization, new regularizors have been specifically designed for deep learning. Spectral norm based regularizors (Yoshida and Miyato 2017; Brock, Donahue, and Simonyan 2018; Farnia, Zhang, and Tse 2019; Miyato et al. 2018) are applied to improve performance of deep learning in applications. In (Neyshabur et al. 2018; Bartlett, Foster, and Telgarsky 2017; Golowich, Rakhlin, and Shamir 2018), analysis suggests that parameter with smaller norm leads to tighter generalization bounds. Entropy based regularizor (Chaudhari et al. 2017) is proposed to encourage training algorithm to produce flat minimas, which are believed to generalize well (Chaudhari et al. 2017; Wu, Zhu, and others 2017). While reducing generalization gap, these regularizors also introduce bias to the objective function. With a biased objective function, the optimization result may not achieve the lowest possible empirical risk. As the performance of a model is determined by both the empirical risk and the performance gap, it is preferable to have regularizors introducing minimal or no bias, while achieving small generalization gap. In this work, we aim at such goal and investigate whether a regularizor can *improve the generalization of deep learning model without biasing the loss function?*.

We answer this question by introducing proximal type penalty function to regularize deep net training (Rockafellar 1976; Combettes and Pesquet 2011; Parikh, Boyd, and others 2014). At t-th iteration of the training, the proximal type quadratic penalty $\lambda/2\|w - w_{t-1}\|_2^2$ enforces search region of parameter w to be close to w_{t-1}. It works as a local regularizor so each gradient step is relatively stable. Different from regularizors that directly apply to loss, the proximal type regularizor is only used to stablize each gradient decent step and does not affect the loss or objective function. Therefore the local regularizor serves as an *Unbiased Regularizor*.

The proximal type algorithm has been well applied in machine learning problems (Xiao and Zhang 2014; Shalev-Shwartz and Zhang 2014; Lin, Mairal, and Harchaoui 2015; Li et al. 2014; 2014). As a quadratic function, the proximal regularizor improves the convexity of loss surface, and thus accelerates optimization algorithms (Lin, Mairal, and Harchaoui 2015; Allen-Zhu 2018a; 2018b; Paquette et al. 2018; Nesterov 2012). Recently, researchers have revisited such regularizor in non-convex optimization context, due to its power in convexifying loss surface and improving optimization efficiency. However, so far, little is known about its generalization power.

In this paper, we study local regularized optimization algorithm which applies proximal type regularizor to stabilize training process. Specifically, we focus on methods that combines such local regularizor with SGD (called LRSGD) and prove *for the first time* that local regularizor can indeed improve generalization bound of SGD. Our analysis relies on the algorithmic stability property of a training algorithm, i.e, the prediction of a trained model does not change much when one training sample is switched. Such stability of a learning

*equal contribution

algorithm was first introduced by Bousquet and Elisseeff (Bousquet and Elisseeff 2002) to show that minimizer of a regularized loss is uniformly stable. Elisseeff et al. extend the algorithmic stability to randomized algorithms (Elisseeff, Evgeniou, and Pontil 2005). Hardt et al. (Hardt, Recht, and Singer 2016) analyzed the stability of iterative algorithms using a progressive divergence argument.

Our analysis show locally regularized SGD is stable and indeed minimizes empirical loss no slower than SGD. Particularly, LRSGD has an inner loop SGD to minimize the regularized objective function. We analyze two variants of LRSGD by setting the number of iterations of the inner loop SGD to a random number (called LRSGD-R) or a constant (called LRSGD-C). These two methods complement each other; one is more stable and the other converges faster. We provide the following theoretical results.

1. LRSGD-R has a better stability (Theorem 4.4) while maintaining the same convergence rate as SGD (Lemma 4.6) which lead to better generalization bound than SGD,.
2. LRSGD-C has significantly less dependence on variance of gradients than SGD, thus converges faster (Lemma 4.10). Meanwhile, LRSGD-C has the same amount of stability guarantee per iteration as SGD (Theorem 4.9), thus generalizes better than SGD.

Our theoretical results are supported by experiments. We observe consistently better generalization performance of LRSGD-R and LRSGD-C over SGD on different neural net architectures. Local regularization has many desirable properties and can be combined with state of the art optimization algorithms e.g., Adam (Kingma and Ba 2014), AdaGrad(Duchi, Hazan, and Singer 2011). It has been and will continue being exploited to design novel training algorithms.

Outline. Section 2 introduces basic notations, assumptions on loss functions and optimization methods. Section 3 contains known results about SGD and our new stability bound for SGD, which is tighter in reasonable parameter regime. In Section 4, analysis about stability and convergence results for both LRSGD-R and LRSGD-C are presented. In section 5, we provide experimental evidence to support our theoretical results. Due to space constraints, we only present simplified theorems, leaving full theorems and proofs to the supplemental material.

2 Preliminaries

This section begins with some classic conditions for analyzing optimization algorithms (Nesterov 2013; Allen-Zhu 2018a). The following conditions regulate the behavior of 0-th order, $1st$ order and $2nd$ order derivatives of $f(x)$.

Condition 2.1 (C2.1)**.** A twice differentiable function $f(x)$ is

- ℓ-Lipschitz smooth, or equivalently ℓ-Hessian bounded if $\|\nabla f(x) - \nabla f(y)\| \leq \ell\|x - y\|$.
- G-Lipschitz continuous, or equivalently G-gradient bounded if $\|f(x) - f(y)\| \leq G\|x - y\|$.
- B-bounded if $|f(x)| \leq B$.

Next we introduce the *non-convexity coefficient* δ, which measures how 'non-convex' a function is.

Algorithm 1 SGD

Input: $h(w), M, \eta, w'$
Initialization $w^0 = w'$
for $k = 1, \cdots, M$ **do**
  Randomly pick $I_k \in [n]$ samples
  $w^{k+1} = w^k - \eta^k \nabla h_{I^k}(w^k)$
end for
Return w^M

Algorithm 2 LRSGD

Input: T, $f(x)$, λ
Initialize w_0
for $t = 1, \cdots, T$ **do**
  $G_t(w) = f(w) + \frac{\lambda_t}{2}\|w - w_t\|^2$
  $w_{t+1} = SGD(G_t(w), M_t, \eta_t, w_t)$
end for
Return w_T

Condition 2.2 (C2.2: δ-non-convex)**.** A function $f(x)$ is δ-non-convex if $f(x) \geq f(y) + \langle \nabla f(y), x - y\rangle - \frac{\delta}{2}\|x - y\|^2$.

Note that δ is universally bounded by the smooth parameter ℓ. We use $\kappa = \delta/\ell \leq 1$ as an *inverse condition number* in the paper. One can also show that the minimal eigenvalue of the Hessian of $f(x)$ is lower bounded by $-\delta$. We next demonstrate the SGD and LRSGD algorithms:

SGD The SGD takes $h(w)$ the object function, M the number of iterations, η the learning rate and w' the initialization as input. Denote by I_k a mini-batch sample set randomly picked in the k-th iteration to compute the stochastic gradient of a loss function $h(w)$, $\nabla h_{I^k}(w^k) = \frac{1}{|I^k|}\sum_{z_j \in I^k} \nabla h_j(w^k, z_j)$. Throughout this paper, w^k denotes the parameter obtained by SGD, η and α denote step size interchangeably upon different conditions.

Next condition regulates the behavior of the stochastic gradients. This is crucial for stochastic gradient to be informative instead of being dominated by noise (Rakhlin et al. 2012; Allen-Zhu 2018a).

Condition 2.3 (C2.3: σ^2-variance-bounded stochastic gradients)**.** Let $\sigma^k = \nabla h_{I^k}(w^k) - \nabla h(w^k)$ be the difference between the stochastic gradient and the exact gradient. We say $h(x)$ has σ^2-variance-bounded stochastic gradients if $\|\sigma^k\|^2 \leq \sigma^2$ (or σ-variance-bounded, in short).

LRSGD An LRSGD also takes gradient descent steps. In each iteration, instead of directly computing the gradient of the loss $\nabla f(w)$, it solves an auxiliary loss function $G_t(w) = f(w) + \frac{\lambda_t}{2}\|w - w_t\|^2$ using an inner loop SGD. The inner loop SGD algorithm minimizes the augmented loss function $G_t(w)$ with M_t number of iterations and η_t learning rate. The output of the inner loop SGD is the next gradient update, w_{t+1}. There are several parameters: the penalty weight λ_t, the SGD learning rate η_t and the number of iterations of SGD M_t. A family of LRSGD algorithms can be derived by setting them differently, which include SGD ($\lambda_t = 0$, $M_t = 1$).

In general, people prefer setting λ_t to be bigger than the non-convexity coefficient δ, so that the auxiliary function

$G_t(w)$ becomes convex, and is easier to solve. *Note that by replacing SGD in inner loop, the framework of LRSGD can be adopt to other training algorithm e.g. Adam (Kingma and Ba 2014).*

When λ_t is non-zero, it plays a role as an inverse learning rate $\lambda_t = 1/\alpha'_t$ for the outer loop update. This is because the update rule could be re-written as $w_{t+1} = w_t - \alpha'_t \nabla f(w_{t+1})$ if one can get an exact solution achieving first order stationary point of $G_t(w)$ in each iteration. In rest of the papers, $\alpha'_t = 1/\lambda_t$ represents step size of LRSGD and η_t denotes learning rate in inner loop SGD. In addition, $\eta_t^k, k \in [M_t], t \in [T]$ are used to represent learning rate in k-th iteration of inner loop SGD and t-th iteration of outer loop LRSGD.

3 Convergence and Stability of SGD

In this section we define uniform stability and relate it to the generalization gap. In Sections 3 and 3, we review known results and adopt them into our notations. In Section 3 we prove a new stability bound of SGD based on the non-convexity coefficient δ. The new bound can be tighter than the existing one in reasonable parameter regime. Another purpose of this new result is: similar analysis can be applied to LRSGD, making it easier to compare the stabilities of the two algorithms.

Convergence of SGD

The following known lemma (Allen-Zhu 2018a; Ge et al. 2015; Reddi et al. 2016) describes a lower bound on how much progress is made in each iteration in expectation. In the lemma, α_t represents the step size of SGD (also known as *learning rate*), which controls the magnitude of the step toward the negative gradient direction. We use α instead of η (as used in Algorithm 1) for ease of comparison with Lemma 4.6 and 4.10 in Section 4.

Lemma 3.1 (One round convergence of SGD)**.** *Assume Conditions C2.1 and C2.3 hold, by picking step size $\alpha_t \leq 1/(2\ell)$ in each round SGD has the following progress:*

$$\mathbb{E}\left[f(w_t) - f(w_{t+1})\right] \geq \mathbb{E}\left[\frac{\alpha_t}{2}\|\nabla f(w_t)\|^2\right] - \alpha_t \sigma^2$$

Remark 3.2. If one uses the exact gradient (the variance bound of gradient $\sigma = 0$), by picking $\alpha_t = 1/(2\ell)$, one can achieve the well-known $O(\ell B/T)$ convergence result (Nesterov 2013). In Lemma 3.1, the variance term σ^2 which comes from noise of stochastic gradient slows down the convergence. For large σ, the SGD step is not guaranteed to decrease function loss. This side effect of SGD is remedied in Section 4 by a scheme of LRSGD which improves the generalization in a 'train faster' fashion.

Stability and Generalization Bound of SGD

In this section we present known stability result of SGD, We begin with reviewing the concept of stability and its connection with *generalization gap*, i.e., the difference between the empirical risk and the population risk. (Bousquet and Elisseeff 2002; Elisseeff, Evgeniou, and Pontil 2005; Hardt, Recht, and Singer 2016). Denote by $S_1 = \{x_1, ..., x_n\}$ the training set with n samples, and S_2 be a perturbed twin of S_1 by replacing the i-th observation x_i with x'_i. Throughout this paper, we denote by v and w the parameters trained using S_1 and S_2 respectively.

The *empirical risk* of parameter w on a training set S is: $R_S(w) = \frac{1}{n}\sum_{i=1}^n f(w; x_i)$, and the *population risk* of parameter w is: $R(w) = \mathbb{E}_x[f(w; x)]$. We denote by $A(\cdot)$ a randomized optimization algorithm (SGD or LRSGD), which employs the iterative update rule. We say $A(\cdot)$ is *uniformly stable* if the change of the output model is bounded when a sample in training set is switched. Formally:

Definition 3.3 (ϵ-uniform randomized stability)**.** A randomized algorithm $A(\cdot)$ is ϵ-uniformly stable if for two datasets of size n, $S_1 = \{x_1, \ldots, x_i, \ldots, x_n\}$ and its perturbed twin $S_2 = \{x_1, \ldots, x'_i, \ldots, x_n\}$, we have

$$\sup_x \mathbb{E}_A[|f(A(S_1); x) - f(A(S_2); x)|] \leq \epsilon,$$

where x represents a test sample from population. Equivalently, we may say the *stability* of $A(\cdot)$ is ϵ.

One can apply a symmetric argument to show that :*ϵ-uniformly stable property of $A(\cdot)$ implies that the expected generalization gap of $A(\cdot)$ is bounded by ϵ* (Bousquet and Elisseeff 2002; Hardt, Recht, and Singer 2016). Therefore, a tight stability bound implies a tight generalization bound. *For the rest of the paper, we will focus on stability bounds, instead of generalization bounds.* We conclude this section by recalling the known stability result of SGD from (Hardt, Recht, and Singer 2016) with slight revision. We incorporate the non-convexity coefficient δ into the original stability result to derive below Theorem.

Theorem 3.4. *Assume Conditions C2.1 and C2.2 hold, SGD with T iterations using step size $\alpha_t = c/(t\ell)$ has stability:*

$$\epsilon^{SGD} = O\left(\frac{1}{n} T^{\frac{c\kappa}{c\kappa+1}} B^{\frac{1}{c\kappa+1}}\right) \tag{1}$$

Remark 3.5. The non-convexity coefficient δ gives a more flexible description of the loss function. Note that above bound depends on the inverse condition number $\kappa = \delta/\ell$. The fact that $\delta \leq \ell$ makes above bound slightly better than original statement in (Hardt, Recht, and Singer 2016) (Theorem 3.12). When $\delta = \ell$, the theorem becomes the original result for SGD. Details of such revision is included in supplementary materials.

A New Stability Bound for SGD

We prove a new stability bound for SGD (Theorem 3.6) which is complimentary to the known result (Theorem 3.4). Compared with Theorem 3.4, our new theorem eliminates the dependence on function value, and thus achieving a better bound for large value of B ($B > \log^2(n)$). This new bound is also easier to compare with the stabilities of LRSGD, as we will prove.

We stress that the proof is nontrival and can be found in the supplemental material.

Theorem 3.6. *Assume Conditions C2.1 and C2.2 hold, SGD with $T(T > n)$ iterations using step size $\alpha_t = c/(t\ell)$ has stability:*

$$\epsilon^{SGD} = O\left(\frac{\log(n) T^{c\kappa}}{n^{1+c\kappa}}\right) = \widetilde{O}\left(\frac{T^{c\kappa}}{n^{1+c\kappa}}\right) \tag{2}$$

Remark 3.7. Note that for the bound in Theorem 3.4 to be nontrivial we need $\frac{1}{n}T^{\frac{c\kappa}{c\kappa+1}}B^{\frac{1}{1+c\kappa}} \leq 2B$ as $\epsilon^{SGD} = \mathbb{E}[|f(w;z) - f(v;z)|]$ is uniformly bounded by $2B$. This automatically gives $\frac{1}{n}T^{\frac{c\kappa}{c\kappa+1}} \leq 2B^{\frac{c\kappa}{c\kappa+1}}$. Consider the bound in Theorem 3.6: $\tilde{O}\big((\frac{1}{n}T^{\frac{c\kappa}{c\kappa+1}})^{c\kappa+1}\big)$, it removes the dependence on B with additional $c\kappa$ on the power number. From the ratio of two bounds one can derive that if $c\kappa \leq \frac{1}{3}$ and $B \geq \log^2(n)$, Theorem 3.6 has a tighter bound than Theorem 3.4. Note that $B \geq \log^2(n)$ is a classic setting in regression problems with quadratic loss.

4 Convergence and Stability of LRSGD

In this section we analyze the stability of LRSGD and compare it with SGD. We start by formalizing LRSGD and its relationship with existing algorithms. In Section 4 and 4, we propose two schemes of LRSGD by setting the number of iterations of the inner loop SGD to a random number(LRSGD-R) or a constant(LRSGD-C). We show LRSGD could either improve the progress made in each iteration (LRSGD-C) or help reduce the divergence quantity in each iteration thus make the training algorithm more stable (LRSGD-R). We leave proof details to supplemental material.

LRSGD-R: A More Stable Algorithm

We define the first scheme LRSGD-R by setting the number of iterations of the inner loop SGD, M_t, to a random number. In each iteration, M_t generated from a geometric distribution $Geo(\gamma)$ with success probability γ, namely, $Pr(M_t = K) = (1-\gamma)^{K-1}\gamma$. For ease of analysis, we introduce an additional parameter θ which fully determines γ. We will later show how θ controls a balance between stability and convergence of LRSGD-R. Formally,

Definition 4.1 (LRSGD-R)**.** LRSGD-R is Algorithm 2 with the following setting: step size in inner loop SGD $\eta_t = \theta/\lambda_t$ where $\theta \leq 1/4$, number of iterations $M_t \sim Geo(\gamma)$ where $\gamma = \frac{log(1/(1-\theta))}{log(1/\theta)}$.

Remark 4.2. Setting the number of iterations M_t to be a Geometric random variable could significantly simplify the analysis in convergence, so that we can explicitly compare the progress made in each iteration with SGD. This trick first appears in (Lei et al. 2017; Lei and Jordan 2017) to control stochastic variance reduction algorithm. The choice of success rate of Geometric distribution γ is proper. Given $\theta \leq 1/4$, $\frac{\theta^2}{1-\theta} \leq \frac{log(1/(1-\theta))}{log(1/\theta)} \leq \frac{\theta}{(1-\theta)^2} \leq 2\theta$. As one can see, θ controls the upper and lower bounds of γ thus plays a crucial role in generating M_t. $\mathbb{E}[M_t] = \frac{1}{\gamma} \geq \frac{1}{2\theta}$, smaller θ results in larger M_t in expectation.

Analysis of LRSGD-R In this section we show that LRSGD-R achieves better generalization performance. In each iteration, LRSGD-R reduces the same loss as SGD, while the stabilities deteriorate in a better rate. Thus when both LRSGD-R and SGD reach a sufficiently low empirical risk, the former generalizes better.

Stability LRSGD-R: Following theorems state that LRSGD-R achieves better algorithmic stability compared to SGD (Theorem 3.4 and 3.6). The key idea of the proof is by applying the strongly convexity of regularized loss function to bound the one round divergence of LRSGD-R and carefully balancing the hyper-parameters. We leave the details in supplementary material.

The following theorem is a direct comparison with Theorem 3.4 of SGD.

Theorem 4.3 (Stability of LRSGD-R)**.** *Assume Conditions C2.1 and C2.2 hold, let* $t_0 = (2(1-\theta)cG^2/\ell)^{1/(q+1)}T^{q/q+1}), q = (1-\theta)c\kappa$. *If one run SGD with step size* $\alpha_t = c/(t\ell)$ *for first* t_0 *rounds then apply LRSGD-R for* $T - t_0$ *iterations with* $\lambda_t = ct\ell$, *one can achieve following stability:*

$$\epsilon^{\text{LRSGD-R}} = O\left(\frac{1}{n}T^{\frac{(1-\theta)c\kappa}{1+(1-\theta)c\kappa}}B^{\frac{1}{1+(1-\theta)c\kappa}}\right) \tag{3}$$

The following theorem is a direct comparison with Theorem 3.6 of SGD.

Theorem 4.4 (Stability of LRSGD-R)**.** *Assume Conditions C2.1 and C2.2 hold, LRSGD-R with* $T(T > n)$ *iteration using parameter* $\lambda_t = ct\ell$ *has stability:*

$$\epsilon^{\text{LRSGD-R}} = \tilde{O}\left(\frac{T^{(1-\theta)c\kappa}}{n^{1+(1-\theta)c\kappa}}\right) \tag{4}$$

Remark 4.5. Compared with the SGD bound in Theorem 3.6, the stability bound in Theorem 4.3 and Theorem 4.4 gives approximately $1-\theta$ improvement on the power number of T. We may set θ to up to $1/4$ to have the tightest stability bound. However, there is a catch. As we will show later, bigger θ will slow down the convergence.

Convergence of LRSGD-R We next show LRSGD-R achieves a convergence rate comparable to SGD (Lemma 3.1). We compare the lower bound of improvement made by SGD and LRSGD in each iteration. Following lemma states that LRSGD-R converges no slower than SGD.

Lemma 4.6. *(one round convergence of LRSGD-R) Assume Conditions C.2.1, C.2.2 and C.2.3 hold, and* $\mathbb{E}[\cdot]$ *is taken over the random variable from Geometric distribution and the stochastic gradient oracle, we have following inequality by picking* $\gamma \leq 2\theta$, $\lambda_t \geq 2\ell$ *and* $\theta = \lambda_t\eta_t \leq 1/4$*:*

$$\mathbb{E}\left[f(w_t) - f(w_{t+1})\right] \geq \frac{\alpha_t'}{2}\mathbb{E}\left[\|\nabla f(w_{t+1})\|^2\right] - \alpha_t'\theta\sigma^2$$

Similar to Lemma 3.1, we can prove a convergence guarantee by telescoping the Lemma 4.6 for LRSGD-R. Unfortunately, we can not apply a classical fixed step size analysis to explicitly derive iteration complexity bound due to necessity of decreasing step size in stability analysis. However, we stress that by comparing with Lemma 3.1, we have a fair comparison with SGD.

Remark 4.7. The convergence rate is not slowed down as α_t in Lemma 3.1 has the same scale as α_t' by setting $\lambda_t = t\ell$. Indeed, the dependence of the bound on the variance of gradient in Theorem 4.6 is improved by a factor of θ.

A balance between stability and convergence. Note that θ adjusts the balance between stability improvement and convergence improvement over SGD. When $\theta \to 0$, the stability improvement brought by local regularizor vanishes. But the algorithm converges with a faster rate than SGD, as the variance term in the RHS of Equation (4.6), $(-\alpha'_t\theta\sigma^2)$, increases with $\theta \to 0$. As θ increases, the stability is improved while the convergence rate slows down. When θ reaches its upper bound $1/4$, one achieves only a constant improvement on variance dependence but an approximately $\Omega\left((\frac{T}{n})^{c\kappa/4}\right)$ improvement on stability. A sweet spot might be found in future analysis and careful tuning in experiments. See Section 5 for an experimental study.

Can we do faster? We have seen that by pushing θ to its upper bound, generalization power is maximized. When we push θ to the other end, we alleviated side effect of variance in stochastic optimization and improve the convergence. However, this setting of LRSGD-R is impractical as the expected number of iterations M_t goes to ∞ as θ goes to 0. *This motivates us to design another scheme with maximal convergence rate while maintaining a stability on par with SGD.*

LRSGD-C: Train faster

We propose another scheme, LRSGD-C, by setting the number of iterations of the inner loop SGD M_t to a large enough constant. In this case, we show how to boost convergence by exploiting the strong convexity of $G_t(w)$ in an optimal fashion, yet still has on-par stability increase as SGD for each round. Overall, since LRSGD-C converges with less iterations than SGD, it has a better stability. We use ψ as a precision parameter of LRSGD-C which controls the optimality of the solution of $G_t(w)$ for every $t \in [T]$. In sum, smaller ψ (and better optimization) demands a larger number of iterations M_t in the inner loop SGD, as a price for accurate solution.

Definition 4.8. LRSGD-C is LRSGD with the following setting: pick the inner SGD learning rate and number of iterations to be $\eta_t^k = \frac{1}{(\lambda_t-\delta)k}$ and $M_t \geq \frac{4\ell^2(G^2+\sigma^2)}{\psi^2(\lambda_t-\delta)}$, respectively. Here $k \in [M_t]$ represents the k-th gradient step within the inner loop SGD. $t \in [T]$ represents the t-th outer loop iteration of LRSGD-C. ψ is the aforementioned precision parameter. G and δ^2 represent the norm upper bound and variance of stochastic gradient.

Since $G_t(w)$ is $(\lambda_t - \delta)$-strongly convex, one can choose a $1/(k(\lambda_t - \delta))$ step size to achieve the optimal convergence rate $O(1/M_t)$ for $G_t(w)$ in stochastic setting (Rakhlin et al. 2012; Nemirovski et al. 2009). Thus we use inner loop SGD to achieve a precise solution to approximate the exact proximal type update $w_{t+1} = w_t - \alpha'_t\nabla f(w_{t+1})$ up to ψ error. The noise coming from stochastic gradient can thus be avoided.

Analysis of LRSGD-C In this section we analyze stability and convergence behavior of LRSGD-C. We show that in each iteration, LRSGD-C reduces significantly more loss than SGD, while their stabilities deteriorate in a same rate. Therefore, LRSGD-C requires much fewer iterations to achieve a sufficiently low empirical risk. Thus LRSGD-C loses much less stability compared with SGD and generalizes better.

Stability LRSGD-C: In following theorems we show LRSGD-C achieves the same algorithmic stability compared to SGD (Theorem 3.4) with a proper choice of parameter setting.

Theorem 4.9. *Assume Conditions C2.1 and C2.2 hold, let* $t_0 = (2cG^2/\ell)^{1/(q+1)}T^{q/q+1}, q = c\kappa$. *If one run SGD with step size* $\alpha_t = c/(t\ell)$ *for first* t_0 *rounds then apply LRSGD-C for* $T - t_0$ *iterations with* $\lambda_t = ct\ell$, *one can achieve the same stability as SGD:*

$$\epsilon^{\text{LRSGD-C}} = O\left(\frac{1}{n}T^{\frac{c\kappa}{c\kappa+1}}B^{\frac{1}{c\kappa+1}}\right) \tag{5}$$

Now we demonstrate that this setting could significantly improve the convergence progress made in each *outer loop iteration*.

Convergence of LRSGD-C: Similar to Section 4, we compare the convergence progress of SGD with LRSGD-C by comparing progress made in each iteration. We describe the one round convergence behavior of LRSGD-C and compare it with that of SGD (Lemma 3.1).

Lemma 4.10. *(one round convergence of LRSGD-C) Assume Conditions C2.1 ,C2.2 and C2.3 hold. Let* $\alpha'_t = 1/(2\lambda_t)$. *LRSGD-C has the following one round progress :*

$$\mathbb{E}\left[f(w_t) - f(w_{t+1})\right] \geq \frac{\alpha'_t}{2}\mathbb{E}\left[\|\nabla f(w_{t+1})\|^2\right] - \alpha'_t\psi^2$$

Remark 4.11. Compared with rate of SGD in Lemma 3.1, the σ^2 on the RHS has been replaced by ψ^2. A smaller value of ψ is prefered and one can reduce value of ψ by increasing the number of inner loop iterations M_t. Note that bigger M_t will not hurt the convergence rate, which is measured by the number of outer loops iterations.

Better with noisy gradient / small batch-size. Above analysis also suggests that LRSGD-C framework is better at handling noisy gradient, as its one-round progress bound depends on ψ^2 rather than the variance of gradient σ^2. Indeed, we observe that variance will dominate the gradient norm with mini-batch size(see Figure S.1 in supplemental material). The implication is for settings where we have to use small batch-size due to memory constraints, LRSGD will be much more effective, as we will demonstrate in experiments.

5 Experiments

We empirically show the generalization power of LRSGD-R and LRSGD-C. We show that they generalize better than SGD for different network architectures. We also use ablation studies to show (1) how LRSGD-R balances between stability and efficiency; and (2) LRSGD-C is more robust to noisy gradients than SGD, when the batch size is small.

Dataset and network architectures. All experiments are based on the CIFAR10 dataset which consists of 10 classes of 32×32 color images, with 6k images per class (Krizhevsky and Hinton 2009). They are split into train and test sets with 50k and 10k images, respectively. We use similar data augmentations as (He et al. 2016): zero-padding with 4 pixels on

Table 1: Average test accuracies (%) on LeNet, VGG16, and ResNet18 on the CIFAR10 dataset.

	LeNet	VGG16	ResNet18
SGD	75.09 ± 0.55	91.76 ± 0.19	94.78 ± 0.10
LRSGD-R	76.71 ± 0.17	92.66 ± 0.14	95.80 ± 0.08
LRSGD-C	76.50 ± 0.23	92.49 ± 0.22	95.63 ± 0.07

each side and randomly cropping into an image of size 32×32. We also pre-process the images by normalizing them with per-channel mean and standard deviation. We use three deep neural networks, i.e., LeNet (LeCun et al. 1998), VGG16 (Simonyan and Zisserman 2014) and ResNet18 (He et al. 2016). The momentum and weight decay parameters of SGD are set to be 0.9 and 0.0001.

The Generalization Performance of LRSGD

We train the networks using the setting in which SGD achieves state-of-the-art accuracy, i.e., the number of iteration is 13.7e4 (350 epochs), the batch size is 128, and the learning rate is $\alpha = 0.1$ initially and decayed by 10 in iteration 5.8e4 and 9.8e4 (epoch 150 and 250).

For LRSGD-R, we set $\gamma = 0.1$, $\lambda_t = 0.01/\alpha$. For LRSGD-C, $M_t = 10$ and $\lambda_t = 0.01/\alpha$. We run each method for 10 times and report the mean and standard deviation of the test accuracies (in $\%$) of the converged models in Table 1. Compared with SGD, both LRSGD-R and LRSGD-C achieve better generalization performance on all three architectures. In Figure 1, we draw the train and test curves for one out of the 10 runs. We observe all methods get stabilized after the first learning rate decay (iteration 5.8e4). LRSGD perform better than SGD. LRSGD-R is slightly better than LRSGD-C. They are consistent with our theoretical analysis.

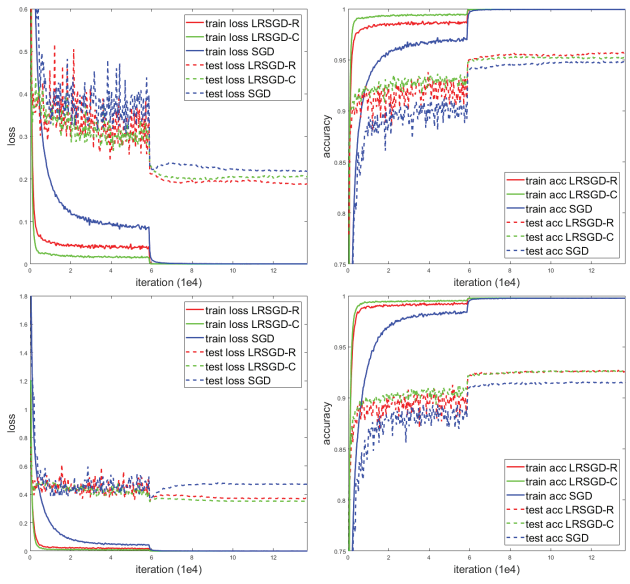


Figure 1: Train and test losses and accuracies on ResNet18 (top two) and VGG16 (bottom two).

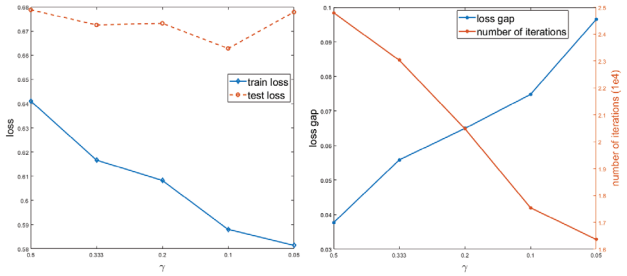


Figure 2: Balance between stability and speed for LRSGD-R: Left: train (blue curve) and test losses (orange curve) as γ increases. Right: hitting time (number of iterations for the training loss to reach 0.65, orange curve) and generalization gap (blue curve) as γ decreases.

LRSGD-R: Balancing Between Stability and Speed

We showed in Section 4 that the paramter θ controls the balance between stability and efficiency of LRSGD-R. In this section, we empirically study how θ affects the performance of LRSGD-R. Note that θ directly determines γ, the success probability of the geometric distribution for generating the number of iterations of the inner loop SGD, M_t. We directly tune the parameter γ and observe how it affects the algorithm from one extreme to another.

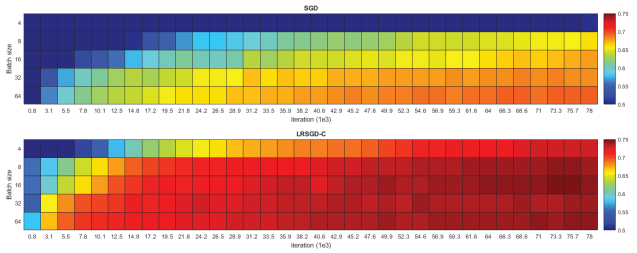
We train the LeNet with LRSGD-R using different values of $\gamma \in [1/2, 1/3, 1/5, 1/10, 1/20]$, corresponding to $\mathbb{E}[M_t] = 2, 3, 5, 10, 20$, respectively. λ_t is set to $0.01/\alpha$. The number of iterations is 3.4e4 (350 epochs) and batch size is 512. The learning rate schedule is the same as that in Section 5. The results are shown in Figure 2. We measure the convergence rate by the hitting time (number of iterations) of LRSGD-R when the training loss reaches a fixed threshold, 0.65. The stability of the algorithm is measured by the generalization gap (test loss - train loss). As γ decreases, the train loss decreases and generalization gap monotonically increases. Meanwhile the hitting time decreases. There is a trade-off between training efficiency and stability, supporting our theoretical analysis in LRSGD-R. In terms of testing accuracy, the optimal choice of γ is 0.1.

LRSGD-C: Robust to Noisy Gradient

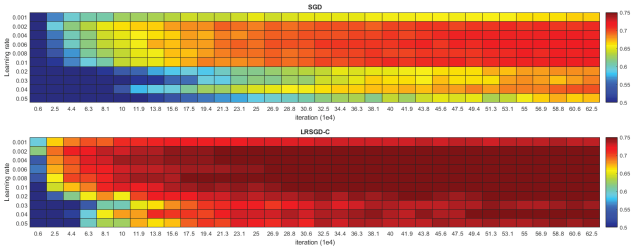
Analysis shows that LRSGD-C is robust to noisy gradient as its one round convergence is controlled by ψ^2 (Lemma 4.10) while SGD's one round convergence is affected by the variance of gradient σ^2 (Lemma 3.1). To validate this analysis, we explore the performance of LRSGD on different small batch sizes. We train the LeNet with LRSGD-C and SGD using batch sizes of $4 \sim 64$, respectively. λ_t is $0.01/\alpha$, $M_t = 10$, the number of iterations is 7.8e4 (1 epoch), and the learning rate of SGD is $\alpha = 0.01$ initially and decayed every 780 iterations. The learning rate of LRSGD-C is $\eta_k = \alpha/k$, where k is the k-th iteration of inner loop SGD. Test accuracies periteration of both SGD and LRSGD-C are presented in Figure 3 (a). LRSGD-C converges to about 76% accuracies for all cases, which is close to the best accuracy (76.50%) using normal batch size (Table 1), while SGD stops at about

71%, which has a larger gap to the result (75.09%) of normal batch size.

It is known that this technique of decreasing step size improves the tolerance of noisy gradients (Rakhlin et al. 2012). A natural question is whether it is this technique or the local regularizer that actually contributes to the robustness to the noisy gradient. To this end, we run a separate experiment on a small batch size of 8, where we fix a constant learning rate $\eta_k = \alpha$ for each inner SGD iteration. Different α values are used to make comprehensive comparison. The test accuracies are shown in Figure 2(b). Values below 0.5 are clamped to 0.5. It is evident that the LRSGD-C achieves better test accuracies than SGD even with fixed inner loop SGD learning rate, proving that the local regularizer is indeed improving the robustness to the noisy gradient.



(a) Different batch sizes



(b) Different learning rates

Figure 3: Test accuracies using (a) different batch sizes and (b) different learning rate: SGD (top) and LRSGD-C (bottom). X-axis: iterations. Y-axis: different batch sizes or learning rates.

6 Conclusion

In this paper we analyze uniform stability of a LRSGD. We theoretically and empirically show that LRSGD framework improves the generalization performance of deep nets. Specifically, we analyze two schemes of LRSGD (LRSGD-R and LRSGD-C). LRSGD-R is more stable than SGD while the convergence is not slow down. We also show that LRSGD-C is as stable as SGD but with a much faster convergence rate (in terms of iterations). We also observed that LRSGD-C can better handle noisy gradient in both analysis and empirical results. This has applications in scenario where small batch sizes has to be used due to the limitation of GPU memory, e.g., in high resolution images synthesis using generative adversarial networks (GANs) (Goodfellow et al. 2014).

Acknowledgements

We thank anonymous reviewers for helpful comments. This work was partially supported by NSF IIS-1855759, CCF-1855760, and CCF-1733843.

References

Allen-Zhu, Z. 2018a. How To Make the Gradients Small Stochastically. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, NeurIPS.

Allen-Zhu, Z. 2018b. Natasha 2: Faster non-convex optimization than sgd. In *Advances in Neural Information Processing Systems*, 2676–2687.

Arora, S.; Ge, R.; Neyshabur, B.; and Zhang, Y. 2018. Stronger generalization bounds for deep nets via a compression approach. In *Proceedings of the 35th International Conference on Machine Learning*, ICML.

Bartlett, P. L.; Foster, D. J.; and Telgarsky, M. J. 2017. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, 6240–6249.

Belkin, M.; Hsu, D.; and Mitra, P. 2018. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, NIPS.

Belkin, M.; Ma, S.; and Mandal, S. 2018. To understand deep learning we need to understand kernel learning. In *Proceedings of the 35th International Conference on Machine Learning*, ICML.

Bousquet, O., and Elisseeff, A. 2002. Stability and generalization. *Journal of machine learning research* 2(Mar):499–526.

Brock, A.; Donahue, J.; and Simonyan, K. 2018. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Brutzkus, A.; Globerson, A.; Malach, E.; and Shalev-Shwartz, S. 2017. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*.

Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J.; Sagun, L.; and Zecchina, R. 2017. Entropy-sgd: Biasing gradient descent into wide valleys. In *ICLR*.

Cohen, N.; Sharir, O.; and Shashua, A. 2016. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, 698–728.

Combettes, P. L., and Pesquet, J.-C. 2011. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*. Springer. 185–212.

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

Elisseeff, A.; Evgeniou, T.; and Pontil, M. 2005. Stability of randomized learning algorithms. *Journal of Machine Learning Research* 6(Jan):55–79.

Farnia, F.; Zhang, J.; and Tse, D. 2019. Generalizable adversarial training via spectral normalization. In *International Conference on Learning Representations*.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2001. *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Ge, R.; Huang, F.; Jin, C.; and Yuan, Y. 2015. Escaping from saddle points–online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, 797–842.

Golowich, N.; Rakhlin, A.; and Shamir, O. 2018. Size-independent sample complexity of neural networks. In *Proceedings of the 31st Conference On Learning Theory*, 297–299.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Hardt, M.; Recht, B.; and Singer, Y. 2016. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, 1225–1234.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hoerl, A. E., and Kennard, R. W. 2000. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* 42(1):80–86.

Kingma, D. P., and Ba, J. L. 2014. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*.

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

Krogh, A., and Hertz, J. A. 1992. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, 950–957.

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

Lei, L., and Jordan, M. 2017. Less than a single pass: Stochastically controlled stochastic gradient. In *Artificial Intelligence and Statistics*, 148–156.

Lei, L.; Ju, C.; Chen, J.; and Jordan, M. I. 2017. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, 2348–2358.

Li, M.; Zhang, T.; Chen, Y.; and Smola, A. J. 2014. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 661–670. ACM.

Lin, H., and Jegelka, S. 2018. Resnet with one-neuron hidden layers is a universal approximator. In *Advances in Neural Information Processing Systems*, 6172–6181.

Lin, H.; Mairal, J.; and Harchaoui, Z. 2015. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, 3384–3392.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Nemirovski, A.; Juditsky, A.; Lan, G.; and Shapiro, A. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization* 19(4):1574–1609.

Nesterov, Y. 2012. How to make the gradients small. *Optima* 88:10–11.

Nesterov, Y. 2013. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

Neyshabur, B.; Bhojanapalli, S.; McAllester, D.; and Srebro, N. 2017. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, 5947–5956.

Neyshabur, B.; Bhojanapalli, S.; McAllester, D.; and Srebro, N. 2018. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *ICLR*.

Paquette, C.; Lin, H.; Drusvyatskiy, D.; Mairal, J.; and Harchaoui, Z. 2018. Catalyst for gradient-based nonconvex optimization. In *AISTATS 2018-21st International Conference on Artificial Intelligence and Statistics*, 1–10.

Parikh, N.; Boyd, S.; et al. 2014. Proximal algorithms. *Foundations and Trends® in Optimization* 1(3):127–239.

Raghu, M.; Poole, B.; Kleinberg, J.; Ganguli, S.; and Dickstein, J. S. 2017. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2847–2854. JMLR. org.

Rakhlin, A.; Shamir, O.; Sridharan, K.; et al. 2012. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, volume 12, 1571–1578. Citeseer.

Reddi, S. J.; Hefny, A.; Sra, S.; Poczos, B.; and Smola, A. 2016. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, 314–323.

Rockafellar, R. T. 1976. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization* 14(5):877–898.

Shalev-Shwartz, S., and Zhang, T. 2014. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International Conference on Machine Learning*, 64–72.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58(1):267–288.

Wu, L.; Zhu, Z.; et al. 2017. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*.

Xiao, L., and Zhang, T. 2014. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization* 24(4):2057–2075.

Yoshida, Y., and Miyato, T. 2017. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2016. Understanding deep learning requires rethinking generalization. *ICLR*.