# Laplacian Smooth Gradient Descent

Stanley J. Osher
Department of Mathematics
University of California, Los Angeles
sjo@math.ucla.edu

Bao Wang
Department of Mathematics
University of California, Los Angeles
wangbaonj@gmail.com

Penhang Yin
Department of Mathematics
University of California, Los Angeles
yph@ucla.edu

Xiyang Luo
Department of Mathematics
University of California, Los Angeles
xylmath@gmail.com

Farzin Barekat
Department of Mathematics
University of California, Los Angeles
fbarekat@math.ucla.edu

Minh Pham
Department of Mathematics
University of California, Los Angeles
minhrose@math.ucla.edu

Alex Lin
Department of Mathematics
University of California, Los Angeles
atlin@math.ucla.edu

April 29, 2019

**Abstract**

We propose a class of very simple modifications of gradient descent and stochastic gradient descent. We show that when applied to a large variety of machine learning problems, ranging from logistic regression to deep neural nets, the proposed surrogates can dramatically reduce the variance, allow to take a larger step size, and improve the generalization accuracy. The methods only involve multiplying the usual (stochastic) gradient by the inverse of a positive definitive matrix (which can be computed efficiently by FFT) with a low condition number coming from a one-dimensional discrete Laplacian or its high order generalizations. It also preserves the mean and increases the smallest component and decreases the largest component. The theory of Hamilton-Jacobi partial differential equations demonstrates that the implicit version of the new algorithm is almost the same as doing gradient descent on a new function which (i) has the same global minima as the original function and (ii) is "more convex". Moreover, we show that optimization algorithms with these surrogates converge uniformly in the discrete Sobolev $H_\sigma^p$ sense and reduce the optimality gap for convex optimization problems. The code is available at: https://github.com/BaoWangMath/LaplacianSmoothing-GradientDescent

## 1 Introduction

Stochastic gradient descent (SGD) [37] has been the workhorse for solving large-scale machine learning (ML) problems. It gives rise to a family of algorithms that enables efficient training of many ML models including deep neural nets (DNNs). SGD utilizes training data very efficiently at the beginning of the training phase, as it converges much faster than GD and L-BFGS during this period [8, 16]. Moreover, the variance of SGD can help gradient-based optimization algorithms circumvent local minima and saddle points and reach those that generalize well [38, 18]. However, the variance of SGD also slows down the convergence after the first few training epochs. To account for the effect of SGD's variance and to ensure the convergence of SGD, a decaying step size has to be applied which is one of the major bottlenecks for the fast convergence of SGD [7, 41, 40]. Moreover, in training many ML models, typically the stage-wise schedule of learning rate is used in practice [39, 38]. In this scenario, the variance of SGD usually leads to a large optimality gap.

A natural question arises from the above bottlenecks of SGD is: **Can we improve SGD such that the variance of the stochastic gradient is reduced on-the-fly with negligible extra computational and memory overhead and a larger step size is allowed to train ML models?**

We answer the above question affirmatively by applying the discrete one-dimensional Laplacian smoothing (LS) operator to smooth the stochastic gradient vector on-the-fly. The LS operation can be performed efficiently by using the fast Fourier transform (FFT). It is shown that the LS reduces the variance of stochastic gradient and allows to take a larger step size.

Another issue of standard GD and SGD is that when the Hessian of the objective function has a large condition number, gradient descent performs poorly. In this case, the derivative increases rapidly in one direction, while growing slowly in another. As a by-product, numerically we will show that LS can avoid oscillation along steep directions and help make progress in shallow directions effectively [25]. The implicit version of our proposed approach is linked to an unusual Hamilton-Jacobi partial differential equation (HJ-PDE) whose solution makes the original loss function more convex while retaining its flat (and global) minima, and essentially works on this surrogate function with a much better landscape. See [10] for earlier, related work.

## 1.1 Our contribution

In this paper, we propose a new modification to the stochastic gradient-based algorithms, which at its core uses the LS operator to reduce the variance of stochastic gradient vector on-the-fly. The (stochastic) gradient smoothing can be done by multiplying the gradient by the inverse of the following circulant convolution matrix

$$\boldsymbol{A}_\sigma := \begin{bmatrix} 1+2\sigma & -\sigma & 0 & \dots & 0 & -\sigma \\ -\sigma & 1+2\sigma & -\sigma & \dots & 0 & 0 \\ 0 & -\sigma & 1+2\sigma & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -\sigma & 0 & 0 & \dots & -\sigma & 1+2\sigma \end{bmatrix} \tag{1}$$

for some positive constant $\sigma \geq 0$. In fact, we can write $\boldsymbol{A}_\sigma = \boldsymbol{I} - \sigma \boldsymbol{L}$, where $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{L}$ is the discrete one-dimensional Laplacian which acts on indices. If we define the (periodic) forward finite difference matrix as

$$\boldsymbol{D}_+ = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ 0 & 0 & -1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & \dots & 0 & -1 \end{bmatrix}.$$

Then, we have $\boldsymbol{A}_\sigma = \boldsymbol{I} - \sigma \boldsymbol{D}_- \boldsymbol{D}_+$, where $\boldsymbol{D}_- = -\boldsymbol{D}_+^\top$ is the backward finite difference.

We summarize the benefits of this simple LS operation below:

- It reduces the variance of stochastic gradient on-the-fly, and reduces the optimality gap when constant step size is used.
- It allows us to take a larger step size than the standard (S)GD.
- It is applicable to train a large variety of ML models including DNNs with better generalization.
- It converges faster for the objective functions that have a large condition number numerically.
- It avoids local sharp minima empirically.

Moreover, as a straightforward extension, we generalize the LS to high-order smoothing operators, e.g., biharmonic smoothing.

## 1.2 Related work

There is an extensive volume of research over the past decades for designing algorithms to speed up the convergence. These include using momentum and other heavy-ball methods, reduce the variance of the stochastic gradient, and adaptive the learning rate. We will discuss the related work from these three perspectives.

The first type of idea to accelerate the convergence of GD and SGD is to apply the momentum. Around local optima, the surface curves can be much more steeply in one dimension than in another [43], whence (S)GD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum. Momentum is proposed to accelerate (S)GD in the relevant direction and dampens oscillations [34]. Nesterov accelerated gradient (NAG) is also introduced to slow down the progress before the surface curve slopes up, and it provably converge faster in specific scenarios [31]. There are lots of recent progress in the development of momentum; a relatively complete survey can be found at [3].

Due to the bottleneck of the variance of the stochastic gradient, a natural idea is to reduce the variance of the stochastic gradient. There are several principles in developing variance reduction algorithms, including Dynamic sample size methods; Gradient aggregation, control variate type of technique is widely used along this direction, some representative works are SAGA [11], SCSG [24], and SVRG [19]; Iterative averaging methods. A thorough survey can be found at [8].

Another category of work tries to speed up the convergence of GD and SGD by using an adaptive step size, which makes use of the historical gradient to adapt the step size. RMSProp [44] and Adagrad [13] adapts the learning rate to the parameters, performing smaller updates (i.e., low learning rates) for parameters associated with frequently occurring features, and more substantial updates (i.e., high learning rates) for parameters associated with infrequent features. Both RMSProp and Adagrad make the learning rate to be historical gradient dependent. Adadelta [48] extends the idea of RMSProp and Adagrad, instead of accumulating all past squared gradients, it restricts the window of accumulated past gradients to some fixed size $w$. Adam [21] and AdaMax [21] behave like a heavy ball with friction, and they compute the decaying averages of past and past squared gradients to adaptive the learning rate. AMSGrad [36] fix the issue of Adam that may fail to converge to an optimal solution. Adam can be viewed as a combination of RMSprop and momentum: RMSprop contributes the exponentially decaying average of past squared gradients, while momentum accounts for the exponentially decaying average of past gradients. Since NAG is superior to vanilla momentum, Dozat [12] proposed NAdam which combines the idea Adam and NAG.

## 1.3 Notations

Throughout this paper, we use boldface upper-case letters $\boldsymbol{A}$, $\boldsymbol{B}$ to denote matrices and boldface lower-case letters $\boldsymbol{w}$, $\boldsymbol{u}$ to denote vectors. For vectors, we use $\|\cdot\|$ to denote the $\ell_2$-norm for vectors and spectral norm for matrices, respectively. And we use $\lambda_{max}(\boldsymbol{A})$, $\lambda_{min}(\boldsymbol{A})$, and $\lambda_i(\boldsymbol{A})$ to denote the largest, smallest, and the $i$-th largest eigenvalues, respectively. For a function $f : \mathbb{R}^n \to \mathbb{R}$, we use $\nabla f$ and $\nabla^2 f$ to denote its gradient and Hessian, and $f^*$ to denote a local minimum of $f$. For a positive definite matrix $\boldsymbol{A}$, we define the vector induced norm by the matrix $\boldsymbol{A}$ as $\|\boldsymbol{w}\|_{\boldsymbol{A}} := \sqrt{\langle \boldsymbol{w}, \boldsymbol{A}\boldsymbol{w}\rangle}$. List $\{1, 2, \cdots, n\}$ is denoted by $[n]$.

## 1.4 Organization

We organize this paper as follows: In section 2, we introduce the LS(S)GD algorithm and the FFT-based fast solver. In section 3, we show that LS(S)GD allows us to take a larger step size than (S)GD based on the and $\ell_2$ estimate of the introduced discrete Laplacian operator. In section 4, we show that LS reduces the variance of SGD both empirically and theoretically. We show that LSGD can avoid some local minima and speed up convergence numerically in section 5. In section 6, we show the benefit of LS in deep learning, including training LeNet [23], ResNet [17], Wasserstein generative adversarial nets (WGAN) [27], and deep reinforcement learning (DRL) model. The convergence analysis for LS(S)GD is provided in section 7. The connection to the Hamilton-Jacobi partial differential equations (HJ-PDEs) and future direction are discussed in section 8. Most of the technical proofs are provided in section 9.

**Algorithm 1** LSSGD

```
Input: $f_i(\boldsymbol{w})$ for $i = 1, 2, \cdots, n$.
$\boldsymbol{w}^0$: initial guess of $\boldsymbol{w}$, $T$: the total number of iterations, and $\eta_k$, $k = 0, 1, \cdots, T$: the scheduled step size.
Output: The optimized weights $\boldsymbol{w}^{\mathrm{opt}}$.
for $k = 0, 1, \cdots, T$ do
    $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \eta \boldsymbol{A}_\sigma^{-1} \left(\nabla f_{i_k}(\boldsymbol{w}^k)\right)$.
return $\boldsymbol{w}^T$
```

## 2 Laplacian Smoothing (Stochastic) Gradient Descent

We present our algorithm for SGD in the finite-sum setting. The GD and other settings follow straightforwardly. Consider the following finite-sum optimization

$$\min_{\boldsymbol{w}} F(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{w}), \tag{2}$$

where $f_i(\boldsymbol{w}) \doteq f(\boldsymbol{w}, \boldsymbol{x}_i, y_i)$ is the loss of a given ML model on the training data $\{\boldsymbol{x}_i, y_i\}$. This finite-sum formalism is an abstract of training many ML models mentioned above. To resolve the optimization problem Eq. (2), starting from some initial guess $\boldsymbol{w}^0$, the $(k+1)$-th iteration of SGD reads

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \eta_k \nabla f_{i_k}(\boldsymbol{w}^k), \tag{3}$$

where $\eta_k$ is the step size, $i_k$ is a random sample with replacement from $[n]$.

We propose to replace the stochastic gradient $\nabla f_{i_k}(\boldsymbol{w}^k)$ by the Laplacian smoothed surrogate, and we call the resulting algorithm LSSGD, which is written as

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \eta_k \boldsymbol{A}_\sigma^{-1} \nabla f_{i_k}(\boldsymbol{w}^k). \tag{4}$$

Intuitively, compared to the standard GD, this scheme smooths the gradient on-the-fly by an elliptic smoothing operator while preserving the mean of the entries of the gradient. We adopt fast Fourier transform (FFT) to compute $\boldsymbol{A}_\sigma^{-1} \nabla f(\boldsymbol{w}^k)$, which is available in both PyTorch [33] and TensorFlow [2]. Given a vector $\boldsymbol{g}$, a smoothed vector $\boldsymbol{d}$ can be obtained by computing $\boldsymbol{d} = \boldsymbol{A}_\sigma^{-1} \boldsymbol{g}$. This is equivalent to $\boldsymbol{g} = \boldsymbol{d} - \sigma \boldsymbol{v} * \boldsymbol{d}$, where $\boldsymbol{v} = [-2, 1, 0, \cdots, 0, 1]^\top$ and $*$ is the convolution operator. Therefore

$$\boldsymbol{d} = \mathrm{ifft}\left(\frac{\mathrm{fft}(\boldsymbol{g})}{\boldsymbol{1} - \sigma \cdot \mathrm{fft}(\boldsymbol{v})}\right),$$

where we use component-wise division (here, fft and ifft are the FFT and inverse FFT, respectively). Hence, the gradient smoothing can be done in quasilinear time. This additional time complexity is almost the same as performing a one step update on the weights vector $\boldsymbol{w}$. For many machine learning models, we may need to concatenate the parameters into a vector. This reshaping might lead to some ambiguity, nevertheless, based on our tests, both row and column majored reshaping work for the LS-GD algorithm. Moreover, in deep learning cases, the weights in different layers might have different physical meanings. For these cases, we perform layer-wise gradient smoothing, instead. We summarize the LSSGD for solving the finite-sum optimization Eq. (2) in Algorithm 1.

**Remark 1.** *In image processing and elsewhere, the Sobolev gradient [20] uses a multi-dimensional Laplacian operator that operates on $\boldsymbol{w}$, and is different from the one-dimensional discrete Laplacian operator employed in our LS-GD scheme that operates on indices.*

It is worth noting that LS is a complement to the heavy ball, e.g., Nesterov momentum, and adaptive learning rate, e.g., Adam, algorithms. It can be combined with these acceleration techniques to speed up the convergence. We will show the performance of these algorithms in the Section 6.

### 2.1 Generalized smoothing gradient descent

We can generalize $\boldsymbol{A}_\sigma$ to the $n$-th order discrete hyper-diffusion operator as follows

$$\boldsymbol{I} + (-1)^n \sigma \boldsymbol{L}^n \doteq \boldsymbol{A}_\sigma^n.$$

Each row of the discrete Laplacian operator $\boldsymbol{L}$ consists of an appropriate arrangement of weights in central finite difference approximation to the 2nd order derivative. Similarly, each row of $\boldsymbol{L}^n$ is an arrangement of the weights in the central finite difference approximation to the $2n$-th order derivative.

**Remark 2.** *The $n$-th order smoothing operator $\boldsymbol{I}+(-1)^n\sigma\boldsymbol{L}^n$ can only be applied to the problem with dimension at least $2n+1$. Otherwise, we need to add dummy variables to the object function.*

Again, we apply FFT to compute the smoothed gradient vector. For a given gradient vector $\boldsymbol{g}$, the smoothed surrogate, $(\boldsymbol{A}_\sigma^n)^{-1}\boldsymbol{g} \doteq \boldsymbol{d}$, can be obtained by solving $\boldsymbol{g} = \boldsymbol{d}+(-1)^n\sigma\boldsymbol{v}_n*\boldsymbol{d}$, where $\boldsymbol{v}_n = (c_{n+1}^n, c_{n+2}^n, \cdots, c_{2n+1}^n, 0, \cdots, 0, c_1^n, c_2^n, \cdots, c_{n-1}^n, c_n^n)$ is a vector of the same dimension as the gradient to be smoothed. And the coefficient vector $\boldsymbol{c}^n = (c_1^n, c_2^n, \cdots, c_{2n+1}^n)$ can be obtained recursively by the following formula

$$\boldsymbol{c}^1 = (1, -2, 1), \quad c_i^n = \begin{cases} 1 & i = 1, 2n+1 \\ -2c_1^{n-1} + c_2^{n-1} & i = 2, 2n \\ c_{i-1}^{n-1} - 2c_i^{n-1} + c_{i+1}^{n-1} & \text{otherwise.} \end{cases}$$

**Remark 3.** *The computational complexities for different order smoothing schemes are the same when the FFT is utilized for computing the surrogate gradient.*

## 3 The Choice of Step Size

In this section, we will discuss the step size issue of LS(S)GD with a theoretical focus on LSGD on $L$-Lipschitz functions.

**Definition 1** ($L$-Lipschitz)**.** *We say the function $F$ is $L$-Lipschitz, if for any $\boldsymbol{w}, \boldsymbol{u} \in \mathbb{R}^m$, we have $\|f(\boldsymbol{w}) - f(\boldsymbol{u})\| \leq L\|\boldsymbol{w} - \boldsymbol{u}\|$.*

**Remark 4.** *If the function $F$ is $L$-Lipschitz and differentiable, then for any $\boldsymbol{w}$, we have $\|\nabla f(\boldsymbol{w})\| \leq L$.*

For $L$-Lipschitz function, it is known that the largest suitable step size for GD is $\eta_{max}^{GD} = \frac{1}{L}$ [32]. In the following, we will establish a $\ell_2$ estimate of the square root of the LS operator when it is applied to an arbitrary vector. Based on these estimates, we will show that LSGD can take a larger step size than GD.

To determine the largest suitable step size for LSGD. We first do a change of variable in the LSGD 2 by letting $\boldsymbol{v}^k = \boldsymbol{H}_\sigma^{-1/2}\boldsymbol{w}^k$ where $\boldsymbol{H}_\sigma = \boldsymbol{A}_\sigma^{-1}$, then LSGD can be written as

$$\boldsymbol{v}^{k+1} = \boldsymbol{v}^k - \eta_k \boldsymbol{H}_\sigma^{1/2}\nabla F(\boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}^k), \tag{5}$$

which is actually the GD for solving the following minimization problem

$$\min_{\boldsymbol{v}} F(\boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}) := \min_{\boldsymbol{v}} G(\boldsymbol{v}). \tag{6}$$

Therefore, to determine the largest suitable step size for LSGD, it is equivalent to find the largest appropriate step size for GD for $\min_{\boldsymbol{v}} G(\boldsymbol{v})$. Therefore, it suffices to determine the Lipschitz constant for the function $G(\boldsymbol{v})$, i.e., to find

$$L_G := \inf_{\boldsymbol{v}} \left\{ \|\nabla G(\boldsymbol{v})\| | \boldsymbol{v} \in \mathrm{dom}(G) \right\}.$$

Note that for $\forall \boldsymbol{v}_1, \boldsymbol{v}_2$, we have

$$\begin{aligned} \|G(\boldsymbol{v}_1) - G(\boldsymbol{v}_2)\| &= \|F(\boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_1) - F(\boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_2)\| \\ &\leq L\|\boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_1 - \boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_2\| \end{aligned}$$

To find the largest appropriate step size, we need to further estimate $\|\boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_1 - \boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_2\|$.

### 3.1 $\ell_2$ estimates of $\mathbf{H}_\sigma^{1/2}\mathbf{v}$

**Proposition 1.** *Given any vector* $\boldsymbol{v} \in \mathbb{R}^m$*, let* $\boldsymbol{w} = \boldsymbol{A}_\sigma^{-1/2}\boldsymbol{v}$*, then*

$$\|\boldsymbol{v}\|^2 = \|\boldsymbol{w}\|^2 + \sigma\|\boldsymbol{D}_+\boldsymbol{w}\|^2. \tag{7}$$

*Proof.* Observe that $\boldsymbol{v} = A_\sigma^{1/2}\boldsymbol{w}$. Therefore,

$$\begin{aligned}\|\boldsymbol{v}\|^2 =& \left\langle \boldsymbol{A}_\sigma^{1/2}\boldsymbol{w}, \boldsymbol{A}_\sigma^{1/2}\boldsymbol{w}\right\rangle = \langle \boldsymbol{A}_\sigma \boldsymbol{w}, \boldsymbol{w}\rangle = \langle \boldsymbol{w} - \sigma\boldsymbol{D}_-\boldsymbol{D}_+\boldsymbol{w}, \boldsymbol{w}\rangle = \|\boldsymbol{w}\|^2 - \sigma\langle \boldsymbol{D}_-\boldsymbol{D}_+\boldsymbol{w}, \boldsymbol{w}\rangle \\ =& \|\boldsymbol{w}\|^2 - \sigma\langle \boldsymbol{D}_+\boldsymbol{w}, -\boldsymbol{D}_+\boldsymbol{w}\rangle = \|\boldsymbol{w}\|^2 + \sigma\|\boldsymbol{D}_+\boldsymbol{w}\|^2,\end{aligned}$$

where we used $\boldsymbol{D}_-^T = -\boldsymbol{D}_+$ for the second last equality. □

Proposition 1 shows that the Lipschitz constant of $G$ is not larger than that of $F$, since

$$\|\boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_1 - \boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_2\|^2 = \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|^2 - \sigma\|\boldsymbol{D}_+(\boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_1 - \boldsymbol{H}_\sigma^{1/2}\boldsymbol{v}_2)\|^2 \leq \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|^2.$$

Therefore, LSGD can take at least the same step size as GD. However, note that $\|\boldsymbol{D}_+\boldsymbol{w}\|_2$ can be arbitrarily close to zero, so LSGD cannot always take a larger step size than GD. Next, we establish a high probability estimation for taking a larger step size when using LSGD.

Without any prior knowledge about $\boldsymbol{v}_1 - \boldsymbol{v}_2 := \boldsymbol{v}$, let us assume it is sampled uniformly from a ball in $\mathbb{R}^m$ centered at the origin. Without loss of generality, we assume the radius of this ball is one. For the sake of notation simplicity, in the following we denote $\boldsymbol{H}_\sigma^{1/2} := \boldsymbol{M}_\sigma$. Under the above ansatz, we have the following result

**Theorem 1** ($\ell_2$-estimate)**.** *Let* $\sigma > 0$*, and*

$$\beta = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{1 + 2\sigma - \sigma z_i - \sigma\overline{z_i}},$$

*where* $z_1, \cdots, z_m$ *are the* $m$ *roots of unity. Let* $\boldsymbol{v}$ *be uniformly distributed in the unit ball of the* $m$ *dimensional* $\ell_2$ *space. Then*

$$\mathbb{P}\left(\|\boldsymbol{M}_\sigma\boldsymbol{v}\| \geq \alpha\|\boldsymbol{v}\|\right) \leq 2\exp\left(-\frac{2}{\pi^2}m\left(\frac{\alpha - \alpha\frac{\pi}{\sqrt{m}} - \sqrt{\beta}}{\alpha + 1}\right)^2\right) \tag{8}$$

*for any* $\alpha > \frac{\sqrt{\beta}}{1 - \frac{\pi}{\sqrt{m}}}$*.*

The proof of this theorem is provided in the appendix. For high dimensional ML problems, e.g., training DNNs, $m$ can be as large as tens of millions so that the probability will be almost one. The closed form of $\beta$ is given in Lemma 1.

**Lemma 1.** *If* $z_1, \ldots, z_m$ *denote the* $m$ *roots of unity, then*

$$\beta = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{1 + 2\sigma - \sigma z_j - \sigma\bar{z}_j} = \frac{1 + \alpha^m}{(1 - \alpha^m)\sqrt{4\sigma + 1}} \to \frac{1}{\sqrt{1 + 4\sigma}}, \tag{9}$$

*as* $m \to \infty$*, where*

$$1 > \alpha = \frac{2\sigma + 1 - \sqrt{4\sigma + 1}}{2\sigma} > 0.$$

The proof of the above lemma requires some tools from complex analysis and harmonic analysis, which is provided in the appendix. Table 1 lists some typical values for different $\sigma$ and dimensions $m$.

Based on the estimate in Theorem 1, LSGD can take the largest step size $\frac{1}{\sqrt{\beta}L}$ for high-dimensional $L$-Lipschitz function with high probability. We will verify this result numerically in the following sections.

Table 1: The values of $\beta$ corresponding to some $\sigma$ and $m$. $\beta$ converges quickly to its limiting value as $m$ increases.

| $\sigma$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $m = 1000$ | 0.447 | 0.333 | 0.277 | 0.243 | 0.218 |
| $m = 10000$ | 0.447 | 0.333 | 0.277 | 0.243 | 0.218 |
| $m = 100000$ | 0.447 | 0.333 | 0.277 | 0.243 | 0.218 |

# 4 Variance Reduction

The variance of SGD is one of the major bottlenecks that slows down the theoretical guaranteed convergence rate in training ML models. Most of the existing variance reduction algorithms require either the full batch gradient or the storage of stochastic gradient for each data point which makes it difficult to be used to train the high-capacity DNNs. LS is an alternative approach to reduce the variance of the stochastic gradient with negligible extra computational time and memory cost. In this section, we rigorously show that LS reduces the variance of the stochastic gradient and reduce the optimality gap under the Gaussian noise assumption. Moreover, we numerically verify our theoretical results on both a quadratic function and a simple finite-sum optimization problem.

## 4.1 Gaussian noise assumption

Stochastic gradient $\nabla f_{i_k}$, for any $i_k \in [n]$, is an unbiased estimate of $\nabla F$, many existing works model the variance between the stochastic gradient and full batch gradient $\nabla F$ as Gaussian noise $\mathcal{N}(\mathbf{0}, \Sigma)$, where $\Sigma$ is the covariance matrix [28]. Therefore, ignoring the variable $\boldsymbol{w}$ for simplicity of notation, we can write the equation involving gradient and stochastic gradient vectors as

$$\nabla f_{i_k} = \nabla F + \boldsymbol{n}, \tag{10}$$

where $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \Sigma)$. Thus for LS stochastic gradient, we have

$$\boldsymbol{A}_\sigma^{-1} \nabla f_{i_k} = \boldsymbol{A}_\sigma^{-1} \left( \nabla F + \boldsymbol{n} \right). \tag{11}$$

The variances of stochastic gradient and LS stochastic gradient are basically the variance of $\boldsymbol{n}$ and $\boldsymbol{A}_\sigma^{-1}\boldsymbol{n}$, respectively. The following theorem quantifies the variance between $\boldsymbol{n}$ and $\boldsymbol{A}_\sigma^{-1}\boldsymbol{n}$.

**Theorem 2.** *Let $\kappa$ denote the condition number of $\Sigma$. Then, for $m$ dimensional Gaussian random vector $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, we have*

$$\frac{\sum_{i=1}^m \mathrm{Var}[\left((\boldsymbol{A}_\sigma^n)^{-1}\boldsymbol{n}\right)_i]}{\sum_{i=1}^m \mathrm{Var}[(\boldsymbol{n})_i]} \leq 1 - \frac{1}{\kappa} + \frac{1}{\kappa m} \sum_{j=0}^{m} \frac{1}{[1 + 4^n \sigma \sin^{2n}(\pi j/m)]^2}. \tag{12}$$

The proof of Theorem 2 will be provided in the appendix.

Table 2 lists the ratio of variance after and before LS for an $m$-dimensional standard normal vector, i.e., $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. In practice, high order smoothing reduce variance more significantly.

Table 2: Theoretical upper bound of $\sum_{i=1}^m \mathrm{Var}[\left((\boldsymbol{A}_\sigma^n)^{-1}\boldsymbol{n}\right)_i] / \sum_{i=1}^m \mathrm{Var}[(\boldsymbol{n})_i]$ when $\mathbf{n}$ is an $m$-dimensional standard normal vector with $m \geq 10000$.

| $\sigma$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $n = 1$ | 0.268 | 0.185 | 0.149 | 0.129 | 0.114 |
| $n = 2$ | 0.279 | 0.231 | 0.207 | 0.192 | 0.181 |
| $n = 3$ | 0.290 | 0.256 | 0.238 | 0.226 | 0.218 |

Moreover, LS preserves the mean (Proposition 2), decreases the largest component and increases the smallest component (Proposition 3) for any vector.

**Proposition 2.** *For any vector $\boldsymbol{g} \in \mathbb{R}^m$, $\boldsymbol{d} = \boldsymbol{A}_\sigma^{-1}\boldsymbol{g}$, let $j_{\max} = \arg\max_i d_i$ and $j_{\min} = \arg\min_i d_i$. We have $\max_i d_i = d_{j_{\max}} \leq g_{j_{\max}} \leq \max_i g_i$ and $\min_i d_i = d_{j_{\min}} \geq g_{j_{\min}} \geq \min_i g_i$.*

*Proof.* Since $\boldsymbol{g} = \boldsymbol{A}_\sigma \boldsymbol{d}$, it holds that

$$g_{j_{\max}} = d_{j_{\max}} + \sigma(2d_{j_{\max}} - d_{j_{\max}-1} - d_{j_{\max}+1}),$$

where periodicity of subindex are used if necessary. Since $2d_{j_{\max}} - d_{j_{\max}-1} - d_{j_{\max}+1} \geq 0$, We have $\max_i d_i = d_{j_{\max}} \leq g_{j_{\max}} \leq \max_i g_i$. A similar argument can show that $\min_i d_i = d_{j_{\min}} \geq g_{j_{\min}} \geq \min_i g_i$. □

**Proposition 3.** *The operator $\boldsymbol{A}_\sigma^{-1}$ preserves the sum of components. For any $\boldsymbol{g} \in \mathbb{R}^m$ and $\boldsymbol{d} = \boldsymbol{A}_\sigma^{-1}\boldsymbol{g}$, we have $\sum_j d_j = \sum_j g_j$, or equivalently, $\mathbf{1}^\top \boldsymbol{d} = \mathbf{1}^\top \boldsymbol{g}$.*

*Proof.* Since $\boldsymbol{g} = \boldsymbol{A}_\sigma \boldsymbol{d}$,

$$\sum_i g_i = \mathbf{1}^\top \mathbf{g} = \mathbf{1}^\top(\boldsymbol{I} + \sigma \boldsymbol{D}_+^\top \boldsymbol{D}_+)\boldsymbol{d} = \mathbf{1}^\top \boldsymbol{d} = \sum_i d_i,$$

where we used $\boldsymbol{D}_+\mathbf{1} = \mathbf{0}$. □

## 4.2 Reduce the optimality gap

A direct benefit of variance reduction is that it reduces the optimality gap in SGD when constant step size is applied. We state the corresponding result in the following.

**Proposition 4.** *Suppose $f$ is convex with the global minimizer $\boldsymbol{w}^*$, and $f^* = f(\boldsymbol{w}^*)$. Consider the following iteration with constant learning rate $\eta > 0$*

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \eta(\boldsymbol{A}_\sigma^n)^{-1}\boldsymbol{g}^k$$

*where $\boldsymbol{g}^k$ is the sampled gradient in the $k$-th iteration at $\boldsymbol{w}^k$ satisfying $\mathbb{E}[\boldsymbol{g}^k] = \nabla f(\boldsymbol{w}^k)$. Denote $G_{\boldsymbol{A}_\sigma^n} := \lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} \|\boldsymbol{g}^k\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}}$ and $\overline{\boldsymbol{w}}^K := \sum_{k=0}^{K-1} \boldsymbol{w}^k/K$ the ergodic average of iterates. Then the optimality gap is*

$$\lim_{K\to\infty} \mathbb{E}[f(\overline{\boldsymbol{w}}^K)] - f^* \leq \frac{\eta G_{\boldsymbol{A}_\sigma^n}}{2}.$$

*Proof.* Since $f$ is convex, we have

$$\langle \nabla f(\boldsymbol{w}^k), \boldsymbol{w}^k - \boldsymbol{w}^* \rangle \geq f(\boldsymbol{w}^k) - f^*. \tag{13}$$

Furthermore,

$$\begin{aligned}
&\mathbb{E}[\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^*\|^2_{\boldsymbol{A}_\sigma^n}] = \mathbb{E}[\|\boldsymbol{w}^k - \eta(\boldsymbol{A}_\sigma^n)^{-1}\boldsymbol{g}^k - \boldsymbol{w}^*\|^2_{\boldsymbol{A}_\sigma^n}] \\
&= \mathbb{E}[\|\boldsymbol{w}^k - \boldsymbol{w}^*\|^2_{\boldsymbol{A}_\sigma^n}] - 2\eta\mathbb{E}[\langle \boldsymbol{g}^k, \boldsymbol{w}^k - \boldsymbol{w}^*\rangle] + \eta^2\mathbb{E}[\|(\boldsymbol{A}_\sigma^n)^{-1}\boldsymbol{g}^t\|^2_{\boldsymbol{A}_\sigma^n}] \\
&\leq \mathbb{E}[\|\boldsymbol{w}^k - \boldsymbol{w}^*\|^2_{\boldsymbol{A}_\sigma^n}] - 2\eta\mathbb{E}[\langle \nabla f(\boldsymbol{w}^k), \boldsymbol{w}^k - \boldsymbol{w}^*\rangle] + \eta^2\|\boldsymbol{g}^k\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}} \\
&\leq \mathbb{E}[\|\boldsymbol{w}^k - \boldsymbol{w}^*\|^2_{\boldsymbol{A}_\sigma^n}] - 2\eta(\mathbb{E}[f(\boldsymbol{w}^k)] - f^*) + \eta^2\|\boldsymbol{g}^k\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}},
\end{aligned}$$

where the last inequality is due to (13). We rearrange the terms and arrive at

$$\mathbb{E}[f(\boldsymbol{w}^k)] - f^* \leq \frac{1}{2\eta}(\mathbb{E}[\|\boldsymbol{w}^k - \boldsymbol{w}^*\|^2_{\boldsymbol{A}_\sigma^n}] - \mathbb{E}[\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^*\|^2_{\boldsymbol{A}_\sigma^n}]) + \frac{\eta\|\boldsymbol{g}^k\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}}}{2}.$$

Summing over $k$ from 0 to $K-1$ and averaging and using the convexity of $f$, we have

$$\mathbb{E}[f(\overline{\boldsymbol{w}}^K)] - f^* \leq \frac{\sum_{k=0}^{K-1}\mathbb{E}[f(\boldsymbol{w}^k)]}{K} - f^* \leq \frac{1}{2\eta K}\mathbb{E}[\|\boldsymbol{w}^0 - \boldsymbol{w}^*\|^2_{\boldsymbol{A}_\sigma^n}] + \frac{\sum_{k=0}^{K-1}\|\boldsymbol{g}^k\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}}}{2K}\eta.$$

Taking the limit as $K \to \infty$ above establishes the result. □

**Remark 5.** *Since $G_{\boldsymbol{A}_\sigma^n}$ is smaller than the corresponding value without LS. It shows that the optimality gap is reduced when LS is used with a constant step size. In practice, this is also true for the stage-wise step size since it is a constant in each stage of the training phase.*

#### 4.2.1 Optimization for quadratic function

In this part, we empirically show the advantages of the LS(S)GD and its generalized schemes for the convex optimization problems. Consider searching the minima $\boldsymbol{x}^*$ of the quadratic function $f(\boldsymbol{x})$ defined in Eq. (14).

$$f(x_1, x_2, \cdots, x_{100}) = \sum_{i=1}^{50} x_{2i-1}^2 + \sum_{i=1}^{50} \frac{x_{2i}^2}{10^2}. \tag{14}$$

To simulate SGD, we add Gaussian noise to the gradient vector, i.e., at any given point $\boldsymbol{x}$, we have

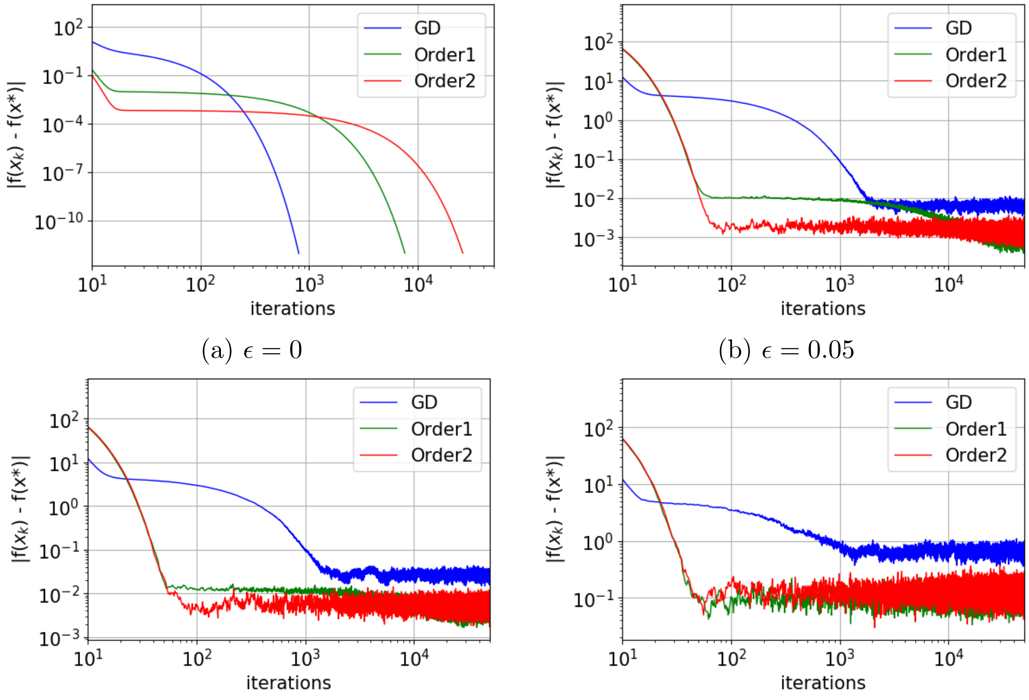
$$\tilde{\nabla}_\epsilon f(\boldsymbol{x}) := \nabla f(\boldsymbol{x}) + \epsilon \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),$$

where the scalar $\epsilon$ controls the noise level, $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is the Gaussian noise vector with zero mean and unit variance in each coordinate. The corresponding numerical schemes can be formulated as

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \eta_k (\boldsymbol{A}_\sigma^n)^{-1} \tilde{\nabla}_\epsilon f(\boldsymbol{x}^k), \tag{15}$$

where $\sigma$ is the smoothing parameter selected to be 10.0 to remove the intense noise. We take diminishing step sizes with initial values 0.1 for SGD/smoothed SGD; 0.9 and 1.8 for GD/smoothed GD, respectively. Without noise, the smoothing allows us to take larger step sizes, rounding to the first digit, 0.9 and 1.8 are the largest suitable step size for GD and smoothed version here. We study both constant learning rate and exponentially decaying learning rate, i.e., after every 1000 iteration the learning rate is divided by 10. We apply different schemes that corresponding to $n = 0, 1, 2$ in Eq. (15) to the problem (Eq. (14)), with the initial point $\boldsymbol{x}^0 = (1, 1, \cdots, 1)$.

Figure. 1 shows the iteration v.s. optimality gap when the constant learning rate is used. In the noise free case, all three schemes converge linearly. When there is noise, our smoothed gradient helps to reduce the optimality gap and converges faster after a few iterations.



(a) $\epsilon = 0$

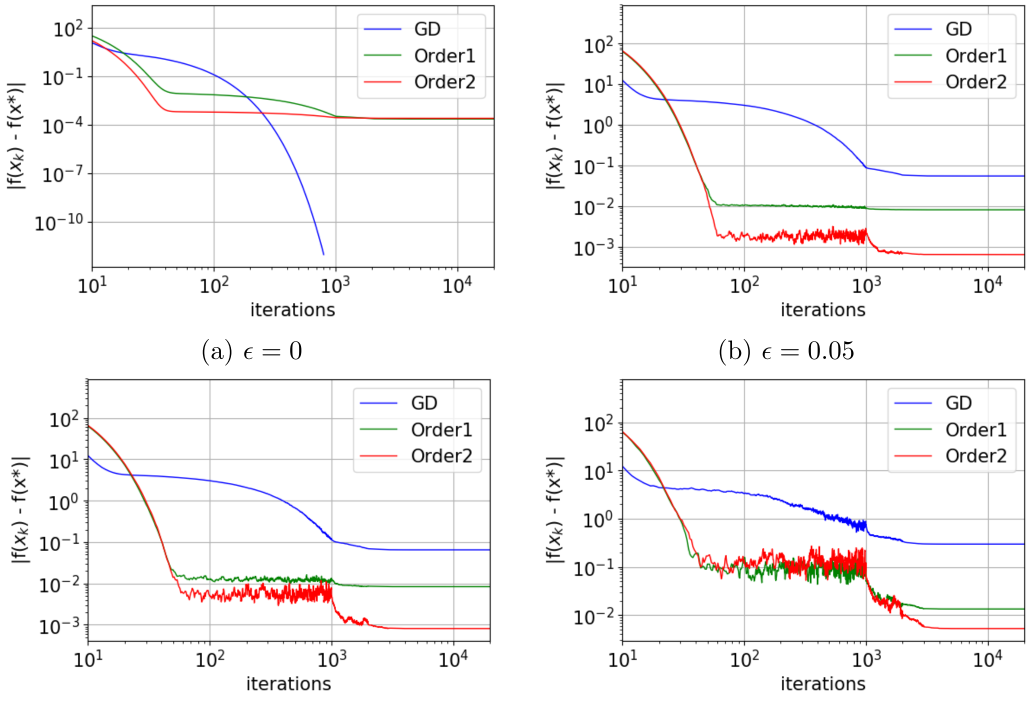
(b) $\epsilon = 0.05$

(c) $\epsilon = 0.1$

(d) $\epsilon = 0.5$

Figure 1: Iterations v.s. optimality gap for GD and smoothed GD with order 1 and order 2 smoothing for the problem in Eq.(14). Constant step size is used.

The exponentially decaying learning rate helps our smoothed SGD to reach a point with a smaller optimality gap, and the higher order smoothing further reduces the optimality gap, as shown in Fig. 2. This is due to the noise removal properties of the smoothing operators.



(a) $\epsilon = 0$

(b) $\epsilon = 0.05$

(c) $\epsilon = 0.1$

(d) $\epsilon = 0.5$

Figure 2: Iterations v.s. optimality gap for GD and smoothed GD with order 1 and 2 smoothing for the problem in Eq.(14). Exponentially decaying step size is utilized here.

### 4.2.2 Find the center of multiple points

Consider searching the center of a given set of 5K random points $\{\mathbf{x}_i \in \mathbb{R}^{50}\}_{i=1}^{5000}$. [1] This problem can be formulate as the following finite-sum optimization

$$\min_{\boldsymbol{x}} F(\boldsymbol{x}) := \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{x}\|^2. \tag{16}$$

We solve this optimization problem by running either SGD or LSSGD for 20K iterations starting from the same random initial point with batch size 20. The initial step size is set to be 1.0 and 1.2, respectively, for SGD and LSSGD, and decays 1.1 times after every 10 iterations. As the learning rate decays, the variance of the stochastic gradient decays [46], thus we decay $\sigma$ 10 times after every 1K iterations. Figure 3 (a) plots a 2D cross section of the trajectories of SGD and LSSGD, and it shows that the trajectory of SGD is more noisy than that of LSSGD. Figure 3 (b) plots the iteration v.s. loss for both SGD and LSSGD averaged over 3 independent runs. LSSGD converges faster than SGD and has a smaller optimality gap than LSSGD. This numerical result verifies our theoretical results on the optimality gap (Proposition 4).

### 4.2.3 Multi-class Logistic regression

Consider applying the proposed optimization sch–emes to train the multi-class Logistic regression model. We run 200 epochs of SGD and different order smoothing algorithms to maximize the likelihood of multi-class Logistic regression with batch size 100. And we apply the exponentially decaying learning rate with initial value 0.5 and decay 10 times after every 50 epochs. We train the model with only 10 % randomly selected MNIST training data and test the trained model on the entire testing images. We further compare with SVRG under the same setting. Figure. 4 shows the histograms of generalization accuracy of the model trained by SGD (a); SVRG (b); LS-SGD (order 1) (c); LS-SGD (oder 2) (d). It is seen that SVRG somewhat improves the generalization with higher averaged accuracy. However, the first and the second order LSSGD type algorithms lift the averaged generalization accuracy by more than 1% and

[1] We thank professor Adam Oberman for suggesting this problem to us.

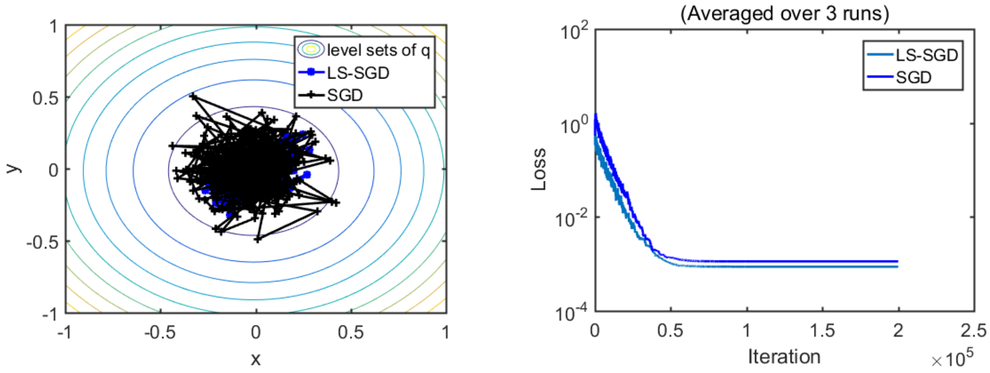


Figure 3: Left: a cross section of the trajectories of SGD and LSSGD. Right: Iteration v.s. Loss for SGD and LS-SGD.

reduce tnt of Electrical Engineering and Computer Sciences University ofhe variance of the generalization accuracy over 100 independent trials remarkably.

### 4.3 Iteration v.s. loss

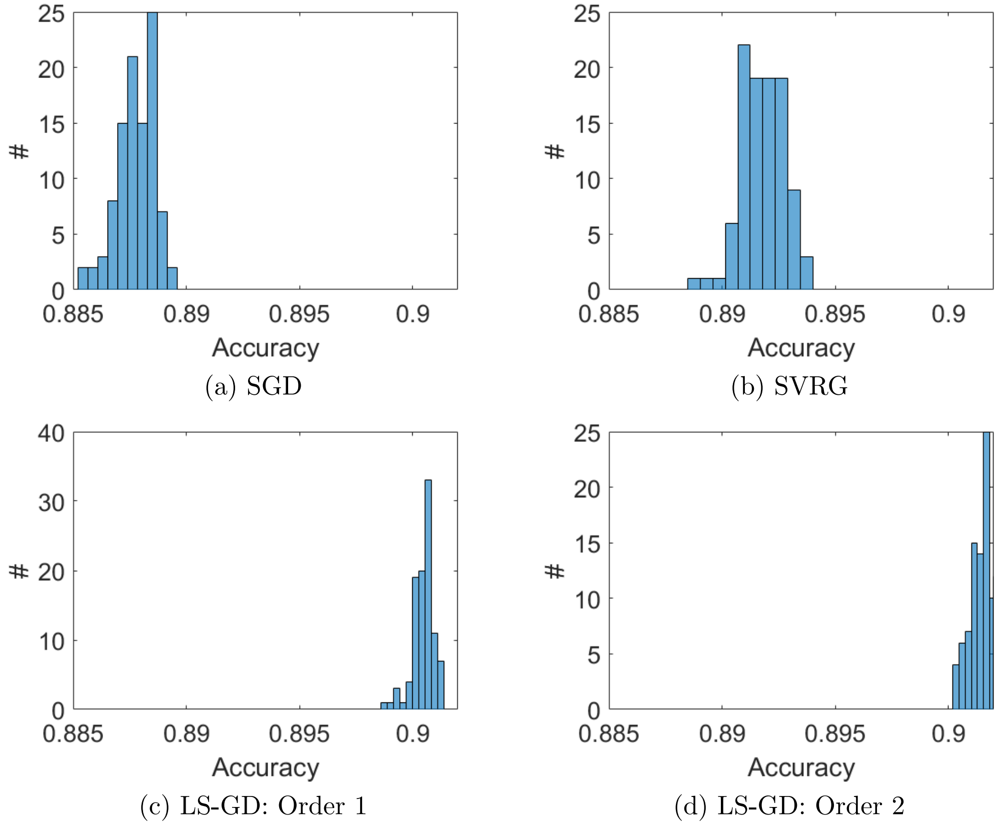
In this part, we show the evolution of the loss in training the multi-class Logistic regression model by SGD, SVRG, LSGD with first and second order smoothing, respectively. As illustrated in Fig. 5. At each iteration, among 100 independent experiments, SGD has the largest variance, SGD with first order smoothed gradient significantly reduces the variance of loss among different experiments. The second order smoothing can further reduce the variance. The variance of loss in each iteration among 100 experiments is minimized when SVRG is used to train the multi-class Logistic model. However, the generalization performance of the model trained by SVRG is not as good as the ones trained by LS-SGD, or higher order smoothed gradient descent (Fig. 4 (b)).

### 4.4 Variance reduction in stochastic gradient

We verify the efficiency of variance reduction numerically in this part. We simplify the problem by applying the multi-class Logistic regression only to the digits 1 and 2 of the MNIST training data. In order to compute the variance of the (LS)-stochastic gradients, we first compute descent path of (LS)-GD by applying the full batch (LS)-GD with learning rate 0.5 starting from the same random initialization. We record the full batch (LS)-gradient on each point along the descent path. Then we compute the (LS)-stochastic gradients on each points along the path by using different batch sizes and smoothing parameters $\sigma$. In computing (LS)-stochastic gradients we run 100 independent experiments. Then we compute the variance of the (LS)-stochastic gradient among these 100 experiments and regarding the full batch (LS)-gradient as the mean on each point along the full batch (LS)-GD descent path. For each pair of batch size and $\sigma$, we report the maximum variance over all the coordinates of the gradient and all the points along the descent path. We list the variance results in Table 3 (note the case $\sigma = 0$ corresponds to the SGD). These results show that compared to the SGD, LSGD with $\sigma = 3$ can reduce the maximum variance $\sim$ **100** times for different batch sizes. It is worth noting that the high order smoothing reduces more variance than the lower order smoothing, this might due to the fact that the noise of SGD is not Gaussian.

Table 3: The maximum variance of the stochastic gradient generated by LS-SGD with different $\sigma$ and batch size. $\sigma = 0$ recovers the SGD.

| Batch Size | 2 | 5 | 10 | 20 | 50 |
|---|---|---|---|---|---|
| $\sigma = 0$ | 1.50E-1 | 5.49E-2 | 2.37E-2 | 1.01E-2 | 4.40E-3 |
| $\sigma = 1$ | 3.40E-3 | 1.30E-3 | 5.45E-4 | 2.32E-4 | 9.02E-5 |
| $\sigma = 2$ | 2.00E-3 | 7.17E-4 | 3.46E-4 | 1.57E-4 | 5.46E-5 |
| $\sigma = 3$ | 1.40E-3 | 4.98E-4 | 2.56E-4 | 1.17E-4 | 3.97E-5 |



(a) SGD

(b) SVRG

(c) LS-GD: Order 1

(d) LS-GD: Order 2

Figure 4: Histogram of testing accuracy over 100 independent experiments of the multi-class Logistic regression model trained on randomly selected 10% MNIST data by different algorithms.

# 5 Numerical Results on Avoid Local Minima and Speed Up Convergence

We first show that LS-GD can bypass sharp minima and reach the global minima. We consider the following function, in which we 'drill' narrow holes on a smooth convex function,

$$f(x,y,z) = -4e^{-\left((x-\pi)^2+(y-\pi)^2+(z-\pi)^2\right)} - \\ 4\sum_i \cos(x)\cos(y)e^{-\beta\left((x-r\sin(\frac{i}{2})-\pi)^2+(y-r\cos(\frac{i}{2})-\pi)^2\right)}, \tag{17}$$

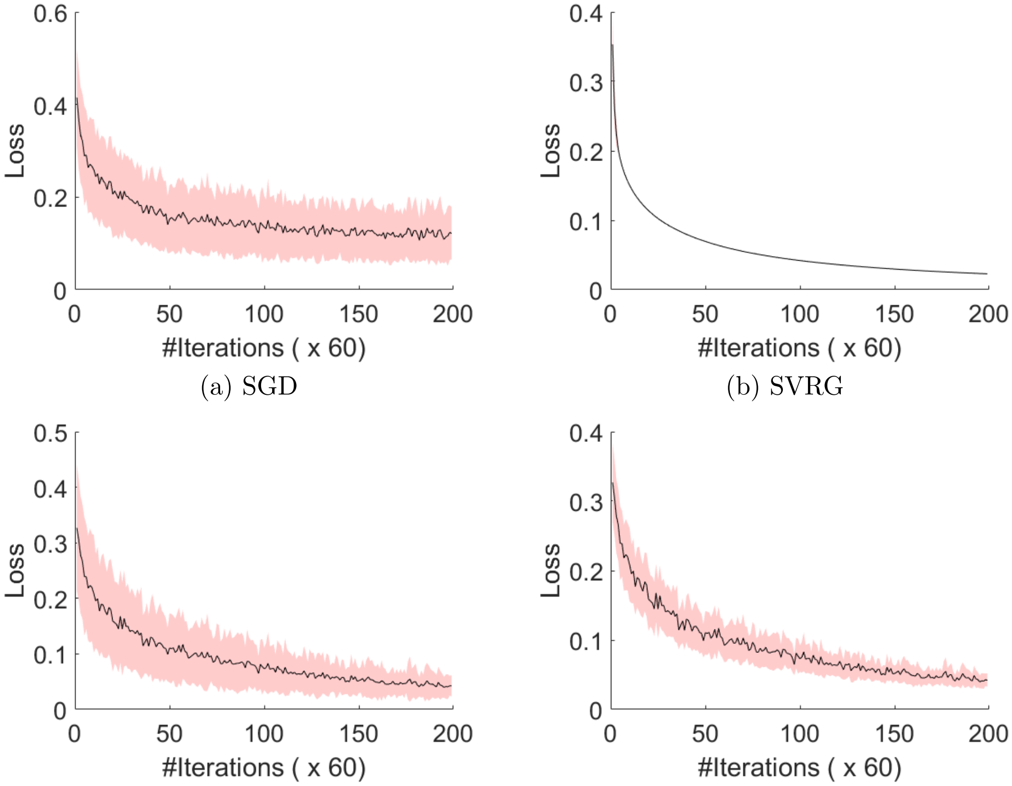
where the summation is taken over the index set $\{i \in \mathbb{N}|\ 0 \leq i < 4\pi\}$, $r$ and $\beta$ are the parameters that determine the location and narrowness of the local minima and are set to 1 and $\frac{1}{\sqrt{500}}$, respectively. We do GD and LS-GD starting from a random point in the neighborhoods of the narrow minima, i.e., $(x_0, y_0, z_0) \in \{\bigcup_i U_\delta(r\sin(\frac{i}{2})+\pi, r\cos(\frac{i}{2})+\pi, \pi)|\ 0 \leq i < 4\pi, i \in \mathbb{N}\}$, where $U_\delta(P)$ is a neighborhood of the point $P$ with radius $\delta$. Our experiments (Fig. 6) show that, if $\delta \leq 0.2$ GD will converge to a narrow local minima, while LS-GD convergences to the wider global minima.

Next, let us compare LSGD with some popular optimization methods on the benchmark 2D-Rosenbrock function which is a non-convex function. The global minimum is inside a long, narrow, parabolic shaped flag valley. To find the valley is trivial. To converge to the global minimum, however, is difficult. The function is defined by

$$f(x,y) = (a-x)^2 + b(y-x^2)^2, \tag{18}$$

it has a global minimum at $(x,y) = (a, a^2)$, and we set $a = 1$ and $b = 100$ in the following experiments.

Starting from the initial point with coordinate $(-3,-4)$, we run 2K iterations of the following optimizers including GD, GD with Nesterov momentum [31], Adam [21], RMSProp [44], and



(a) SGD

(b) SVRG

(c) LS-GD: Order 1

(d) LS-GD: Order 2

Figure 5: Iterations v.s. loss for SGD, SVRG, and LS-SGD with order 1 and order 2 gradient smoothing for training the multi-class Logistic regression model.

LSGD ($\sigma = 0.5$). The step size used for all these methods is $3e-3$. Figure 7 plots the iteration v.s. objective value, and it shows that GD together with Nesterov momentum converges faster than all the other algorithms. The second best algorithm is LSGD. Meanwhile, Nesterov momentum can be used to speed up LSGD, and we will show this numerically in training DNNs in section 6.

Figure 8 depicts some snapshots (The 300th, 600th, 900th, and 1200th iteration, respectively) of the trajectories of different optimization algorithms. These figures show that even though GD with momentum converge faster but it suffers from some overshoots, and they detour to converge to the local minima. All the other algorithms go along a direct path to the minima, and LSGD converges fastest.

Furthermore, we will show that LSGD can be further accelerated by using Nesterov momentum. As show in Fig. 9, the LSGD together with Nesterov momentum converges much faster than GD with momentum, especially for high dimensional Rosenbrock function.

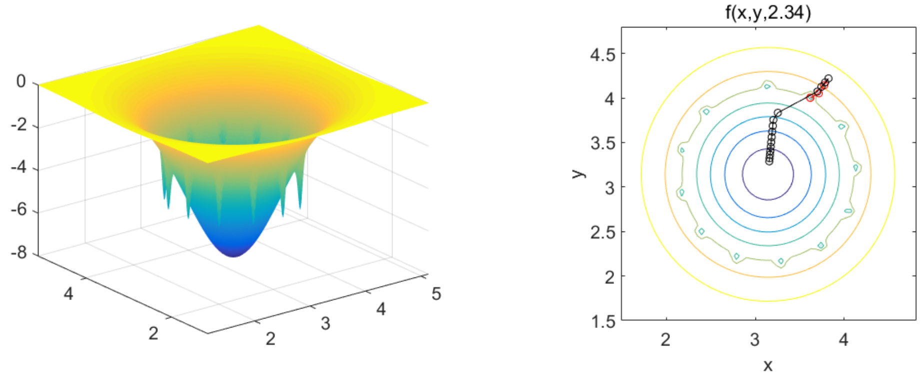
# 6 Application to Deep Learning

## 6.1 Train neural nets with small batch size

Many advanced artificial intelligence tasks make high demands on training neural nets with extremely small batch sizes. The milestone technique for this is group normalization [47]. In this section, we show that LS-SGD successfully trains DNN with extremely small batch size. We consider LeNet-5 [23] for MNIST classification. Our network architecture is as follows

$$\text{LeNet-5: input}_{28\times 28} \rightarrow \text{conv}_{20,5,2} \rightarrow \text{conv}_{50,5,2} \rightarrow \text{fc}_{512} \rightarrow \text{softmax}.$$

The notation $\text{conv}_{c,k,m}$ denotes a 2D convolutional layer with $c$ output channels, each of which is the sum of a channel-wise convolution operation on the input using a learnable kernel of size $k \times k$, it further adds ReLU nonlinearity and max pooling with stride size $m$. $\text{fc}_{512}$ is an affine



(a) (b)

Figure 6: Demo of GD and LS-GD. Panel (a) depicts the slice of the function (Eq.(17)) with $z = 2.34$; panel (b) shows the paths of GD (red) and LS-GD (black). We take the step size to be 0.02 for both GD and LS-GD. $\sigma = 1.0$ is utilized for LS-GD.

transformation that transforms the input to a vector of dimension 512. Finally, the tensors are activated by a multi-class Logistic function. The MNIST data is first passed to the layer $\text{input}_{28\times 28}$, and further processed by this hierarchical structure. We run 100 epochs of both SGD and LS-SGD with initial learning rate 0.01 and divide by 5 after 50 epochs, and use a weight decay of 0.0001 and momentum of 0.9. Figure. 10(a) plots the generalization accuracy on the test set with the LeNet5 trained with different batch sizes. For each batch size, LS-SGD with $\sigma = 1.0$ keeps the testing accuracy more than 99.4%, SGD reduce the accuracy to 97% when batch size 4 is used. The classification become just a random guess, when the model is trained by SGD with batch size 2. Small batch size leads to large noise in the gradient, which may make the noisy gradient not along the decent direction; however, Lapacian smoothing rescues this by decreasing the noise.

## 6.2 Improve generalization accuracy

The skip connections in ResNet smooth the landscape of the loss function of the classical CNN [17, 26]. This means that ResNet has fewer sharp minima. On Cifar10 [22], we compare the performance of LS-SGD and SGD on ResNet with the pre-activated ResNet56 as an illustration. We take the same training strategy as that used in [17], except that we run 200 epochs with the learning rate decaying by a factor of 5 after every 40 epochs. For ResNet, instead of applying LS-SGD for all epochs, we only use LS-SGD in the first 40 epochs, and the remaining training is carried out by SGD (this will save the extra computational cost due to LS, and we noticed that the performance is similar to the case when LS is used for the whole training process). The parameter $\sigma$ is set to 1.0. Figure 10(b) depicts one path of the training and generalization accuracy of the neural nets trained by SGD and LS-SGD, respectively. It is seen that, even though the training accuracy obtained by SGD is higher than that by LS-SGD, the generalization is however inferior to that of LS-SGD. We conjecture that this is due to the fact that SGD gets trapped into some sharp but deeper minimum, which fits better than a flat minimum but generalizes worse. We carry out 25 replicas of this experiments, the histograms of the corresponding accuracy are shown in Fig. 11.

## 6.3 Training Wassersterin GAN

Generative Adversarial Networks (GANs) [15] are notoriously delicate and unstable to train [4]. In [27], Wasserstein-GANs (WGANs) are introduced to combat the instability in the training GANs. In addition to being more robust in training parameters and network architecture, WGANs provide a reliable estimate of the Earth Mover (EM) metric which correlates well with the quality of the generated samples. Nonetheless, WGANs training becomes unstable with a large learning rate or when used with a momentum based optimizer [27]. In this section, we demonstrate that the gradient smoothing technique in this paper alleviates the instability in the training, and improves the quality of generated samples. Since WGANs with weight clipping are typically trained with RMSProp [44], we propose replacing the gradient $g$ by a smoothed

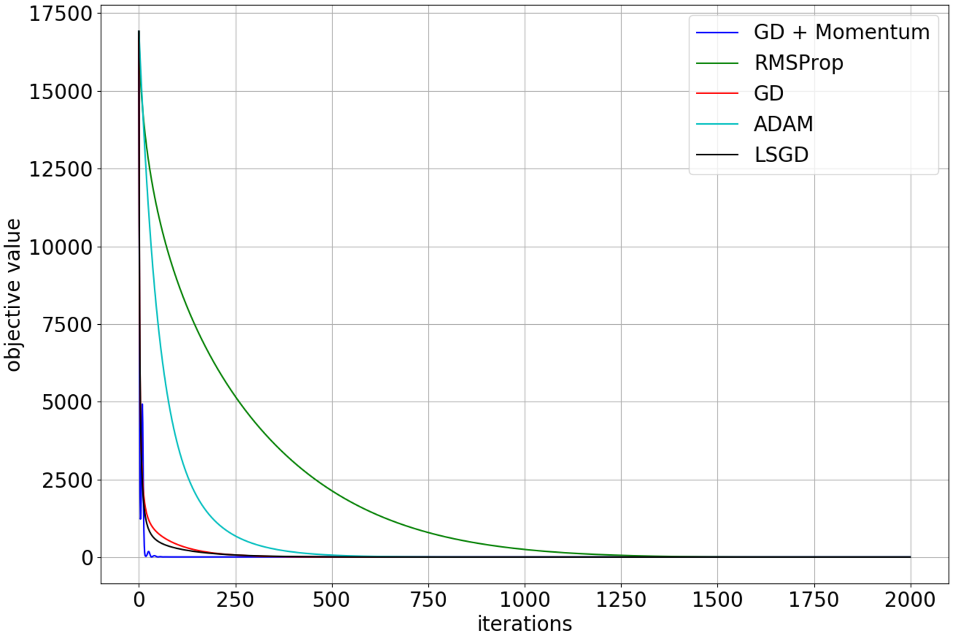


Figure 7: Iteration v.s. loss of different optimization algorithms in optimize the Rosenbrock function.

version $g_\sigma = \boldsymbol{A}_\sigma^{-1} g$, and also update the running averages using $g_\sigma$ instead of $g$. We name this algorithm LS-RMSProp.

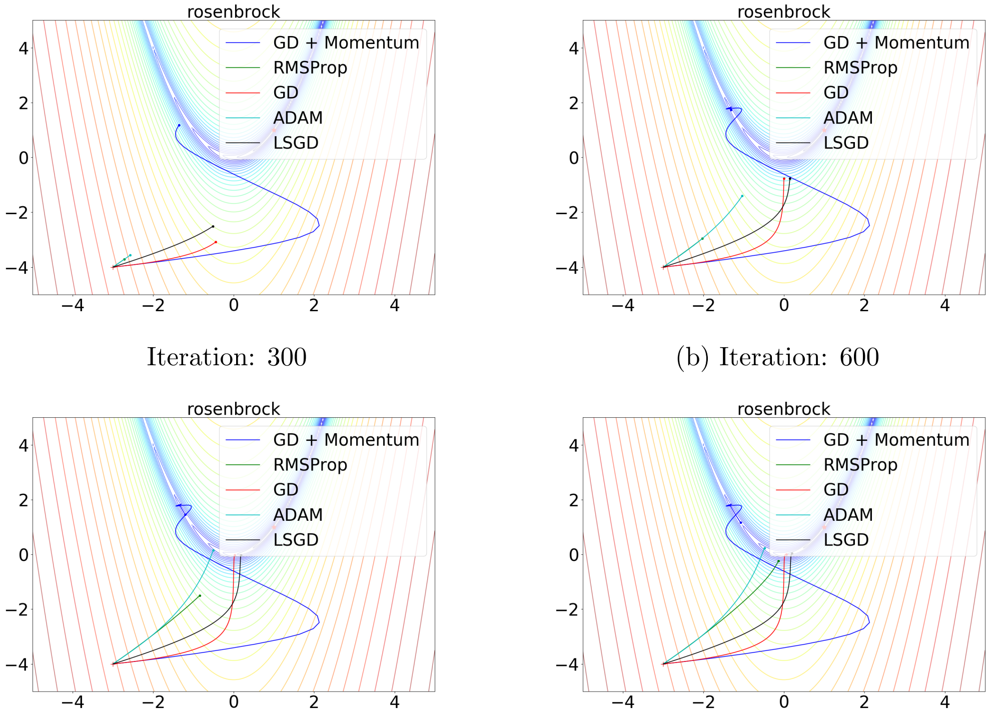
To accentuate the instability in training and demonstrate the effects of gradient smoothing, we deliberately use a large learning rate for training the generator. We compare the regular RMSProp with the LS-RMSProp. The learning rate for the critic is kept small and trained approximately to convergence so that the critic loss is still an effective approximation to the Wasserstein distance. To control the number of unknowns in the experiment and make a meaningful comparison using the critic loss, we use the classical RMSProp for the critic, and only apply LS-RMSProp to the generator.

We train the WGANs on the MNIST dataset using the DCGAN [35] for both the critic and generator. In Figure 12 (top), we observe the loss for RMSProp trained with a large learning rate has multiple sharp spikes, indicating instability in the training process. The samples generated are also lower in quality, containing noisy spots as shown in Figure 13 (a). In contrast, the curve of training loss for LS-RMSProp is smoother and exhibits fewer spikes. The generated samples as shown in Fig. 13 (b) are also of better quality and visibly less noisy. The generated characters shown in Fig. 13 (b) are more realistic compared to the ones shown in Fig. 13 (a). The effects are less pronounced with a small learning rate, but still result in a modest improvement in sample quality as shown in Figure 13 (c) and (d).We also apply LS-RMSProp for training the critic, but do not see a clear improvement in the quality. This may be because the critic is already trained near optimality during each iteration, and does not benefit much from gradient smoothing.

## 6.4 Deep reinforcement learning

Deep reinforcement learning (DRL) has been applied to playing games including Cartpole [9], Atari [30], Go [42, 29]. DNN plays a vital role in approximating the Q-function or policy function. We apply the Laplacian smoothed gradient to train the policy function to play the Cartpole game. We apply the standard procedure to train the policy function by using the policy gradient [9]. And we use the following network to approximate the policy function:

$$\text{input}_4 \rightarrow \text{fc}_{20} \rightarrow \text{relu} \rightarrow \text{fc}_2 \rightarrow \text{softmax}.$$



Iteration: 300

(b) Iteration: 600

Iteration: 900

(d) Iteration: 1200

Figure 8: Some snapshots of trajectories of different optimization algorithms on the Rosenbrock function.

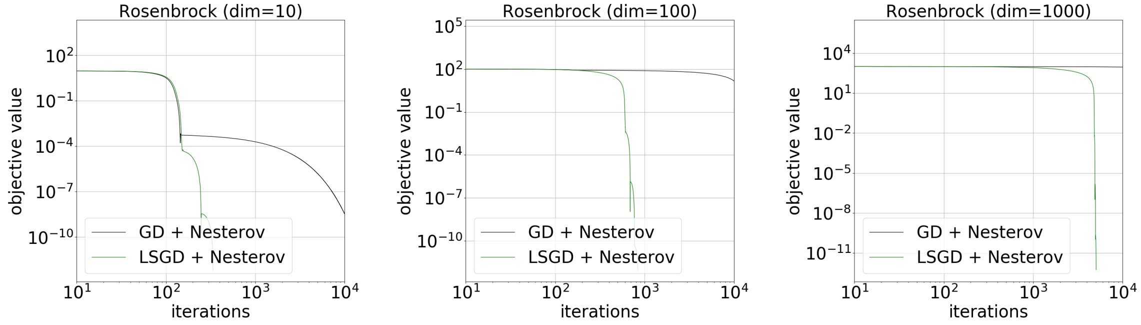


Figure 9: Iteration v.s. objective value for GD with Nesterov momentum and LSGD with Nesterov momentum.

The network is trained by RMSProp and LS-RMSProp with $\sigma = 1.0$, respectively. The learning rate and other related parameters are set to be the default ones in PyTorch. The training is stopped once the average duration of 5 consecutive episodes is more than 490. In each training episode, we set the maximal steps to be 500. Left and right panels of Fig. 14 depict a training procedure by using RMSProp and LS-RMSProp, respectively. We see that Laplacian smoothed gradient takes fewer episodes to reach the stopping criterion. Moreover, we run the above experiments 5 times independently, and apply the trained model to play Cartpole. The game lasts more than 1000 steps for all the 5 models trained by LS-RMSProp, while only 3 of them lasts more than 1000 steps when the model is trained by vanilla RMSProp.

# 7 Convergence Analysis

Note that the LS matrix $\boldsymbol{A}_\sigma^{-1}$ is positive definite and its largest and smallest eigenvalues are 1 and $\frac{1}{1+4\sigma}$, respectively. It is straightforward to show that all the convergence results for (S)GD still hold for LS(S)GD. In this section, we will show some additional convergence for LS(S)GD

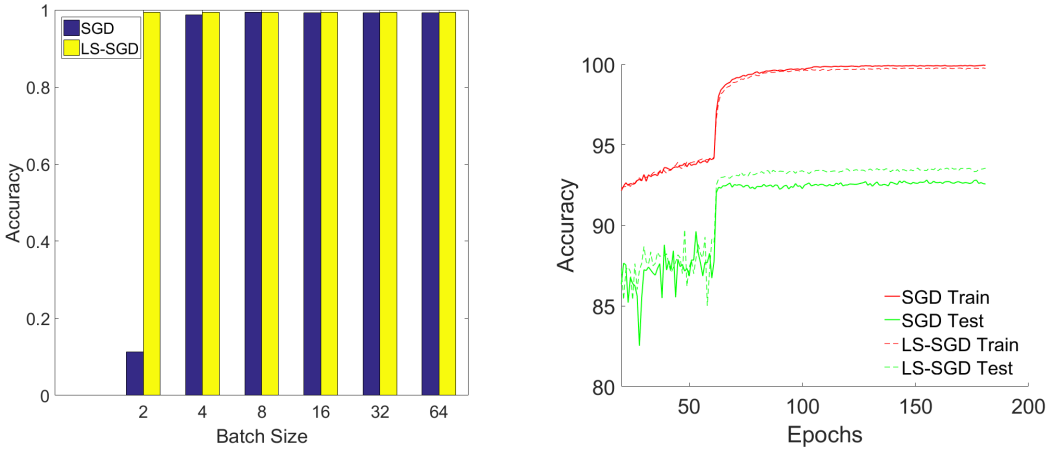


Figure 10: (a). Testing accuracy of LeNet5 trained by SGD/LS-SGD on MNIST with various batch sizes. (b). The evolution of the pre-activated ResNet56's training and generalization accuracy by SGD and LS-SGD. (Start from the 20-th epoch.)

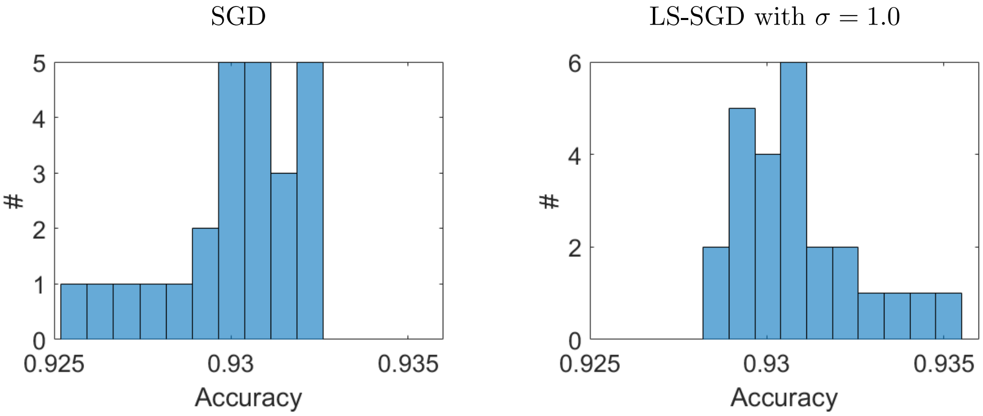


Figure 11: The histogram of the generalization accuracy of the pre-activated ResNet56 on Cifar10 trained by SGD and LS-SGD over 25 independent experiments.

with a focus on LSGD, the corresponding results for LSSGD follow in a similar way.

**Proposition 5.** *Consider the algorithm* $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \eta_k(\boldsymbol{A}_\sigma^n)^{-1}\nabla f(\boldsymbol{w}^k)$. *Suppose* $f$ *is* $L$-*Lipschitz smooth and* $0 < \tilde{\eta} \leq \eta \leq \bar{\eta} < \frac{2}{L}$. *Then* $\lim_{t\to\infty} \|\nabla f(\boldsymbol{w}^k)\| \to 0$. *Moreover, if the Hessian* $\nabla^2 f$ *of* $f$ *is continuous with* $\boldsymbol{w}^*$ *being the minimizer of* $f$, *and* $\bar{\eta}\|\nabla^2 f\| < 1$, *then* $\|\boldsymbol{w}^k - \boldsymbol{w}^*\|_{\boldsymbol{A}_\sigma^n} \to 0$ *as* $k \to \infty$, *and the convergence is linear.*

*Proof.* By the Lipschitz continuity of $\nabla f$ and the descent lemma [5], we have

$$\begin{aligned}
f(\boldsymbol{w}^{k+1}) &= f(\boldsymbol{w}^k - \eta_k(\boldsymbol{A}_\sigma^n)^{-1}\nabla f(\boldsymbol{w}^k)) \\
&\leq f(\boldsymbol{w}^k) - \eta_k\langle \nabla f(\boldsymbol{w}^k), (\boldsymbol{A}_\sigma^n)^{-1}\nabla f(\boldsymbol{w}^k))\rangle + \frac{\eta_k^2 L}{2}\|(\boldsymbol{A}_\sigma^n)^{-1}\nabla f(\boldsymbol{w}^k)\|^2 \\
&\leq f(\boldsymbol{w}^k) - \eta_k\|\nabla f(\boldsymbol{w}^k)\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}} + \frac{\eta_k^2 L}{2}\|\nabla f(\boldsymbol{w}^k)\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}} \\
&\leq f(\boldsymbol{w}^k) - \tilde{\eta}\left(1 - \frac{\bar{\eta}L}{2}\right)\|\nabla f(\boldsymbol{w}^k)\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}}.
\end{aligned}$$

Summing the above inequality over $k$, we have

$$\tilde{\eta}\left(1 - \frac{\bar{\eta}L}{2}\right)\sum_{k=0}^{\infty}\|\nabla f(\boldsymbol{w}^k)\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}} \leq f(\boldsymbol{w}^0) - \lim_{k\to\infty} f(\boldsymbol{w}^k) < \infty.$$

Therefore, $\|\nabla f(\boldsymbol{w}^k)\|^2_{(\boldsymbol{A}_\sigma^n)^{-1}} \to 0$, and thus $\|\nabla f(\boldsymbol{w}^k)\| \to 0$.

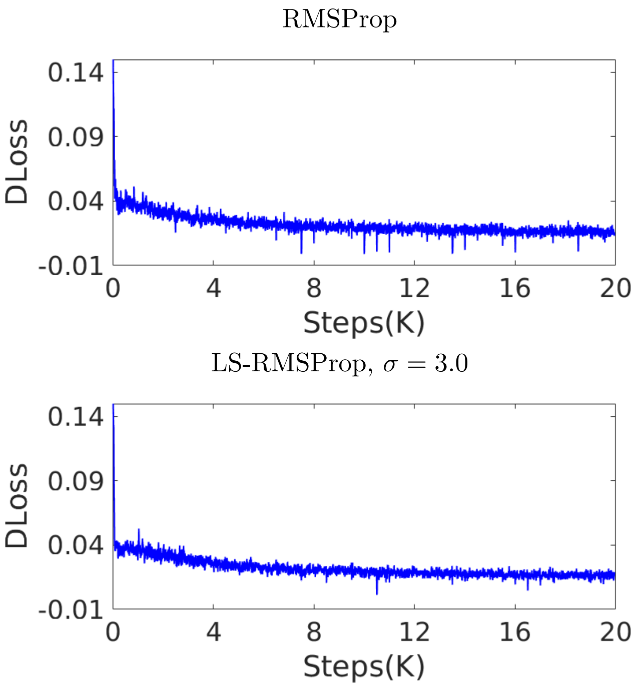
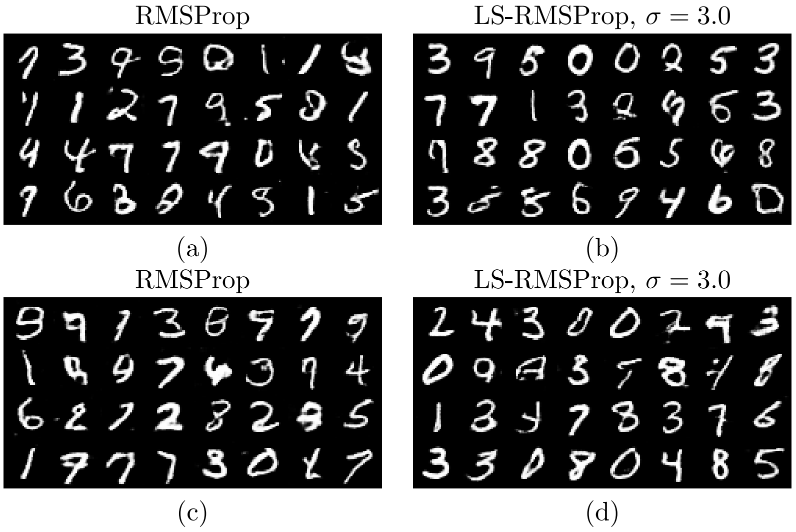


Figure 12: Critic loss with learning rate $lrD = 0.0001$, $lrG = 0.005$ for RMSProp (top) and LS-RMSProp (bottom), trained for 20K iterations. We apply a mean filter of window size 13 for better visualization. The loss from LS-RMSProp is visibly less noisy.



(a) (b) (c) (d)

Figure 13: Samples from WGANs trained with RMSProp (a, c) and LS-RMSProp (b, d). The learning rate is set to $lrD = 0.0001$, $lrG = 0.005$ for both RMSProp and LS-RMSProp in (a) and (b). And $lrD = 0.0001$, $lrG = 0.0001$ are used for both RMSProp and LS-RMSProp in (c) and (d). The critic is trained for 5 iterations per step of the generator, and 200 iterations per every 500 steps of the generator.

For the second claim, we have

$$\begin{aligned}
&\boldsymbol{w}^{k+1} - \boldsymbol{w}^* \\
&= \boldsymbol{w}^k - \boldsymbol{w}^* - \eta_k (\boldsymbol{A}_\sigma^n)^{-1} (\nabla f(\boldsymbol{w}^k) - \nabla f(\boldsymbol{w}^*)) \\
&= \boldsymbol{w}^k - \boldsymbol{w}^* - \eta_k (\boldsymbol{A}_\sigma^n)^{-1} \left( \int_0^1 \nabla^2 f(\boldsymbol{w}^* + \tau(\boldsymbol{w}^{k+1} - \boldsymbol{w}^*)) \cdot (\boldsymbol{w}^k - \boldsymbol{w}^*) \mathrm{d}\tau \right) \\
&= \boldsymbol{w}^k - \boldsymbol{w}^* - \eta_k (\boldsymbol{A}_\sigma^n)^{-1} \left( \int_0^1 \nabla^2 f(\boldsymbol{w}^* + \tau(\boldsymbol{w}^{k+1} - \boldsymbol{w}^*)) \mathrm{d}\tau \cdot (\boldsymbol{w}^k - \boldsymbol{w}^*) \right) \\
&= (\boldsymbol{A}_\sigma^n)^{-\frac{1}{2}} \left( \boldsymbol{I} - \eta_k (\boldsymbol{A}_\sigma^n)^{-\frac{1}{2}} \int_0^1 \nabla^2 f(\boldsymbol{w}^* + \tau(\boldsymbol{w}^{k+1} - \boldsymbol{w}^*)) \mathrm{d}\tau (\boldsymbol{A}_\sigma^n)^{-\frac{1}{2}} \right) (\boldsymbol{A}_\sigma^n)^{\frac{1}{2}} (\boldsymbol{w}^k - \boldsymbol{w}^*)
\end{aligned}$$

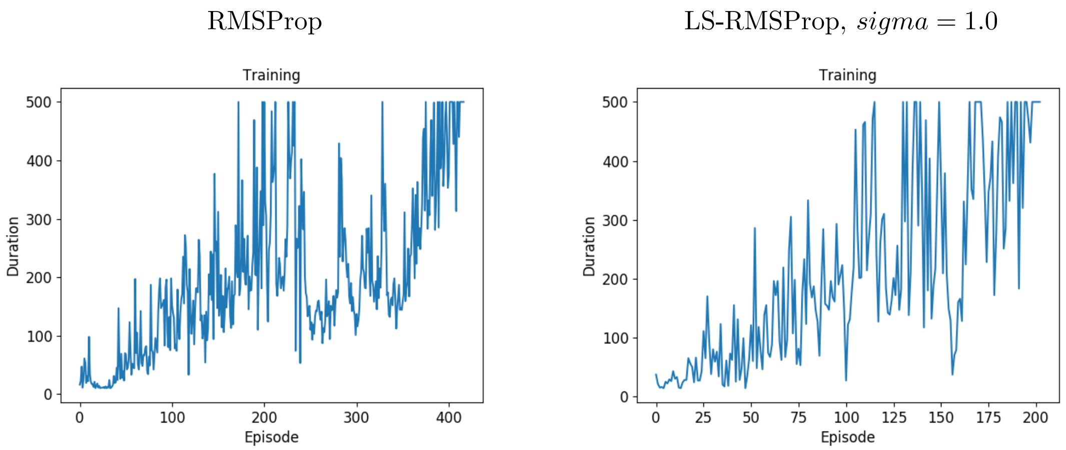


Figure 14: Durations of the cartpole game in the training procedure. Left and right are training procedure by RMSProp and LS-RMSProp with $\sigma = 1.0$, respectively.

Therefore,

$$\|\boldsymbol{w}^{k+1}-\boldsymbol{w}^*\|_{\boldsymbol{A}_\sigma^n} \le \left\|\boldsymbol{I}-\eta_t(\boldsymbol{A}_\sigma^n)^{-\frac{1}{2}}\int_0^1 \nabla^2 f(\boldsymbol{w}^*+\tau(\boldsymbol{w}^{k+1}-\boldsymbol{w}^*))\mathrm{d}\tau(\boldsymbol{A}_\sigma^n)^{-\frac{1}{2}}\right\| \|\boldsymbol{w}^k-\boldsymbol{w}^*\|_{\boldsymbol{A}_\sigma^n}.$$

So if $\eta_k\|\nabla^2 f\| \le \frac{1}{\|(\boldsymbol{A}_\sigma^n)^{-1}\|} = 1$, the result follows. □

**Remark 6.** *The convergence result in Proposition 5 is also call $H_\sigma^n$-convergence. This is because $\langle \boldsymbol{u}, \boldsymbol{A}_\sigma^n\boldsymbol{u}\rangle = \|\boldsymbol{u}\|^2+\sigma\|\boldsymbol{D}_+^n\boldsymbol{u}\|^2 = \|\boldsymbol{u}\|_{H_\sigma^n}^2$.*

# 8 Discussion and Conclusion

## 8.1 Some more properties of Laplacian smoothing

In Theorem 8, we established a high probability estimate of the LS operator in reducing the $\ell_2$ norm of any given vector. The $\ell_1$ type of high probability estimation can be established in the same way. These estimates will be helpful to develop privacy-preserving optimization algorithms to train ML models that improve the utility of the trained models without sacrifice the privacy guarantee [45].

Regarding the $\ell_1/\ell_2$ estimates of the LS operator, we further have the following results.

**Proposition 8.** *Given vectors $\boldsymbol{g}$ and $\boldsymbol{d} = \boldsymbol{A}_\sigma^{-1}\boldsymbol{g}$, for any $p \in \mathbb{N}$, it holds that $\|\boldsymbol{D}_+^p\boldsymbol{d}\|_1 \le \|\boldsymbol{D}_+^p\boldsymbol{g}\|_1$. The inequality is strict unless $\boldsymbol{D}_+^p\boldsymbol{g}$ is a constant vector.*

*Proof.* Observe that $\boldsymbol{A}_\sigma$ and $\boldsymbol{D}_+$ commute; therefore, for any $p \in \mathbb{N}$, $\boldsymbol{A}_\sigma(\boldsymbol{D}_+^p\boldsymbol{d}) = \boldsymbol{D}_+^p\boldsymbol{g}$. Thus we have

$$(1+2\sigma)(\boldsymbol{D}_+^p\boldsymbol{d})_i = (\boldsymbol{D}_+^p\boldsymbol{g})_i + \sigma(\boldsymbol{D}_+^p\boldsymbol{d})_{i+1} + \sigma(\boldsymbol{D}_+^p\boldsymbol{d})_{i-1}.$$

So

$$(1+2\sigma)|(\boldsymbol{D}_+^p\boldsymbol{d})_i| \le |(\boldsymbol{D}_+^p\boldsymbol{g})_i| + \sigma|(\boldsymbol{D}_+^p\boldsymbol{d})_{i+1}| + \sigma|(\boldsymbol{D}_+^p\boldsymbol{d})_{i-1}|.$$

The inequality is strict if there are sign changes among the $(\boldsymbol{D}_+^p\boldsymbol{d})_{i-1}$, $(\boldsymbol{D}_+^p\boldsymbol{d})_i$, $(\boldsymbol{D}_+^p\boldsymbol{d})_{i+1}$. Summing over $i$ and using periodicity, we have

$$(1+2\sigma)\sum_{i=1}^m |(\boldsymbol{D}_+^p\boldsymbol{d})_i| \le \sum_{i=1}^m |(\boldsymbol{D}_+^p\boldsymbol{g})_i| + 2\sigma\sum_{i=1}^m |(\boldsymbol{D}_+^p\boldsymbol{d})_i|,$$

and the result follows. The inequality is strict unless $\boldsymbol{D}_+^p\boldsymbol{g}$ is a constant vector. □

**Proposition 6.** *Given any vector $\boldsymbol{g} \in \mathbb{R}^m$ and $\boldsymbol{d} = (\boldsymbol{A}_\sigma^n)^{-1}\boldsymbol{g}$, then*

$$\|\boldsymbol{g}\|^2 = \|\boldsymbol{d}\|^2 + 2\sigma\|\boldsymbol{D}_+^{\boldsymbol{n}}\boldsymbol{d}\|^2 + \sigma^2\|\boldsymbol{L}^n\boldsymbol{d}\|^2, \tag{19}$$

*the variance of $\boldsymbol{d}$ is much less than that of $\boldsymbol{g}$.*

*Proof.* Observe that $\boldsymbol{g} = \boldsymbol{A}_\sigma^n \boldsymbol{d} = \boldsymbol{d} + (-1)^n \sigma \boldsymbol{L}^n d$. Therefore,

$$\|\boldsymbol{g}\|^2 = \langle \boldsymbol{d} + (-1)^n \sigma \boldsymbol{L}^n \boldsymbol{d}, \boldsymbol{d} + (-1)^n \sigma \boldsymbol{L}^n \boldsymbol{d} \rangle = \|\boldsymbol{d}\|^2 + 2(-1)^n \sigma \langle \boldsymbol{d}, \boldsymbol{L}^n \boldsymbol{d} \rangle + \sigma^2 \|\boldsymbol{L}^n \boldsymbol{d}\|^2. \tag{20}$$

Next, note $\boldsymbol{D}_-$ and $\boldsymbol{D}_+$ are commute; thus

$$\boldsymbol{L}^n = \underbrace{(\boldsymbol{D}_-\boldsymbol{D}_+)\cdots(\boldsymbol{D}_-\boldsymbol{D}_+)}_{n} = \underbrace{\boldsymbol{D}_-\cdots\boldsymbol{D}_-}_{n}\underbrace{\boldsymbol{D}_+\cdots\boldsymbol{D}_+}_{n} = \boldsymbol{D}_-^n \boldsymbol{D}_+^n. \tag{21}$$

Now, we have

$$\langle \boldsymbol{d}, \boldsymbol{L}^n \boldsymbol{d} \rangle = \langle \boldsymbol{d}, \boldsymbol{D}_-^n \boldsymbol{D}_+^n d \rangle = \langle (\boldsymbol{D}_-^n)^T \boldsymbol{d}, \boldsymbol{D}_+^n \boldsymbol{d} \rangle = \langle (-1)^n \boldsymbol{D}_+^n \boldsymbol{d}, \boldsymbol{D}_+^n \boldsymbol{d} \rangle = (-1)^n \|\boldsymbol{D}_+^n \boldsymbol{d}\|^2, \tag{22}$$

where we used Eq. (21) in the first equality and $\boldsymbol{D}_- = -\boldsymbol{D}_+^T$ in the second to last equality.

Substituting Eq. (22) into Eq. (20), yields Eq. (19). □

## 8.2 Connection to Hamilton-Jacobi PDEs

The motivation for the proposed LS-SGD comes from the Hamilton-Jacobi PDE (HJ-PDE). Consider the following unusual HJ-PDE with the empirical risk function, $f(\boldsymbol{w})$, as initial condition

$$\begin{cases} u_t + \frac{1}{2}\big\langle \nabla_{\boldsymbol{w}} u, \boldsymbol{A}_\sigma^{-1} \nabla_{\boldsymbol{w}} u \big\rangle = 0, & (\boldsymbol{w}, t) \in \Omega \times [0, \infty) \\ u(\boldsymbol{w}, 0) = f(\boldsymbol{w}), & \boldsymbol{w} \in \Omega \end{cases} \tag{23}$$

By the Hopf-Lax formula [14], the unique viscosity solution to Eq. (23) is represented by

$$u(\boldsymbol{w}, t) = \inf_{\boldsymbol{v}} \Big\{ f(\boldsymbol{v}) + \frac{1}{2t} \big\langle \boldsymbol{v} - \boldsymbol{w}, \boldsymbol{A}_\sigma (\boldsymbol{v} - \boldsymbol{w}) \big\rangle \Big\}.$$

This viscosity solution $u(\boldsymbol{w}, t)$ makes $f(\boldsymbol{w})$ ”more convex”, an intuitive definition and theoretical explanation of ”more convex” can be found in [10], by bringing down the local maxima while retaining and widening local minima. An illustration of this is shown in Fig. 15. If we perform the smoothing GD with proper step size on the function $u(\boldsymbol{w}, t)$, it is easier to reach the global or at least a flat minima of the original nonconvex function $f(\boldsymbol{w})$.

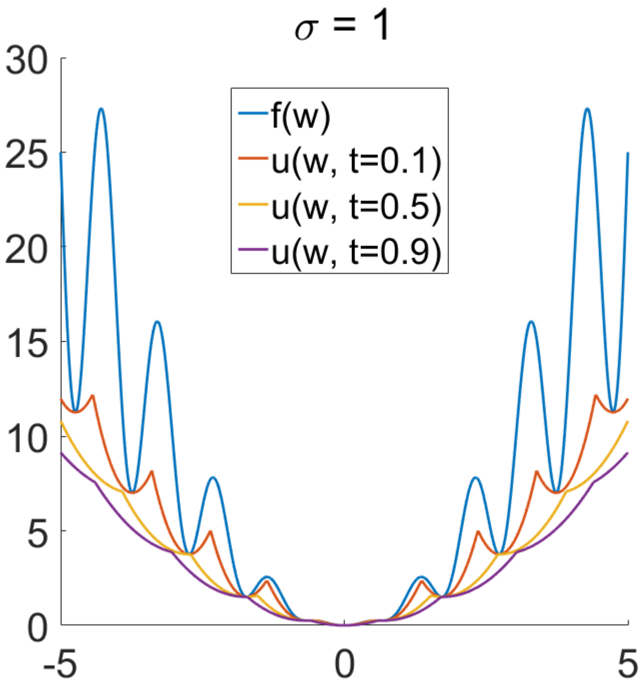


Figure 15: $f(\boldsymbol{w}) = \|\boldsymbol{w}\|^2\big(1 + \frac{1}{2}\sin(2\pi\|\boldsymbol{w}\|)\big)$ is made more convex by solving Eq.(23). The plot shows the cross section of the 5D problem with $\sigma = 1$ and different $t$ values.

**Proposition 1.** *Suppose $f(\boldsymbol{w})$ is differentiable, the LS-GD on $u(\boldsymbol{w},t)$*

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - t\boldsymbol{A}_\sigma^{-1}\nabla_{\boldsymbol{w}} u(\boldsymbol{w}^k, t)$$

*is equivalent to the smoothing implicit GD on $f(\boldsymbol{w})$*

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - t\boldsymbol{A}_\sigma^{-1}\nabla f(\boldsymbol{w}^{k+1}). \tag{24}$$

*Proof.* We define

$$z(\boldsymbol{w},\boldsymbol{v},t) := f(\boldsymbol{v}) + \frac{1}{2t}\langle \boldsymbol{v}-\boldsymbol{w}, \boldsymbol{A}_\sigma(\boldsymbol{v}-\boldsymbol{w})\rangle,$$

and rewrite $u(\boldsymbol{w},t) = \inf_{\boldsymbol{v}} z(\boldsymbol{w},\boldsymbol{v},t)$ as $z(\boldsymbol{w},\boldsymbol{v}(\boldsymbol{w},t),t)$, where $\boldsymbol{v}(\boldsymbol{w},t) = \arg\min_{\boldsymbol{v}} z(\boldsymbol{w},\boldsymbol{v},t)$. Then by the Euler-Lagrange equation,

$$\nabla_{\boldsymbol{w}} u(\boldsymbol{w},t) = \nabla_{\boldsymbol{w}} z(\boldsymbol{w},\boldsymbol{v}(\boldsymbol{w},t),t) = \boldsymbol{J}_{\boldsymbol{w}}\boldsymbol{v}(\boldsymbol{w},t)\nabla_{\boldsymbol{v}} z(\boldsymbol{w},\boldsymbol{v}(\boldsymbol{w},t),t) + \nabla_{\mathbf{w}} z(\boldsymbol{w},\boldsymbol{v}(\boldsymbol{w},t),t),$$

where $\boldsymbol{J}_{\boldsymbol{w}}\mathbf{v}(\boldsymbol{w},t)$ is the Jacobian matrix of $\boldsymbol{v}$ w.r.t. $\boldsymbol{w}$. Notice that $\nabla_{\boldsymbol{v}} z(\boldsymbol{w},\boldsymbol{v}(\boldsymbol{w},t),t) = \mathbf{0}$,

$$\nabla_{\boldsymbol{w}} u(\boldsymbol{w},t) = \nabla_{\boldsymbol{w}} z(\boldsymbol{w},\boldsymbol{v}(\boldsymbol{w},t),t) = -\frac{1}{t}\boldsymbol{A}_\sigma(\boldsymbol{v}(\boldsymbol{w},t)-\boldsymbol{w}).$$

Letting $\boldsymbol{w} = \boldsymbol{w}^k$ and $\boldsymbol{w}^{k+1} = \boldsymbol{v}(\boldsymbol{w}^k,t) = \arg\min_{\mathbf{v}} z(\boldsymbol{w}^k,\boldsymbol{v},t)$ in the above equalities, we have

$$\nabla_{\boldsymbol{w}} u(\boldsymbol{w}^k,t) = -\frac{1}{t}\boldsymbol{A}_\sigma(\boldsymbol{w}^{k+1}-\boldsymbol{w}^k).$$

In summary, the gradient descent $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - t\boldsymbol{A}_\sigma^{-1}\nabla_{\boldsymbol{w}} u(\boldsymbol{w}^k,t)$ is equivalent to the proximal point iteration $\boldsymbol{w}^{k+1} = \arg\min_{\boldsymbol{v}} f(\boldsymbol{v}) + \frac{1}{2t}\langle \boldsymbol{v}-\boldsymbol{w}^k, \boldsymbol{A}_\sigma(\boldsymbol{v}-\boldsymbol{w}^k)\rangle$, which yields $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - t\boldsymbol{A}_\sigma^{-1}\nabla f(\boldsymbol{w}^{k+1})$. □

The studied LS-GD algorithm is an explicit relaxation of the implicit algorithm in Eq.(24).

### 8.3 Conclusion

Motivated by the theory of Hamilton-Jacobi partial differential equations, we proposed Laplacian smoothing gradient descent and its high order generalizations. This simple modification dramatically reduces the variance and optimality gap in stochastic gradient descent, allows us to take a larger step size, and helps to find better minima. Extensive numerical examples ranging from toy cases and shallow and deep neural nets to generative adversarial networks and deep reinforcement learning, all demonstrate the advantage of the proposed smoothed gradient. Several issues remain, in particular devising an on-the-fly adaptive method for choosing the smoothing parameter $\sigma$ instead of using a fixed value.

# 9 Appendix

## 9.1 Proof of Theorem 1

In this part, we will give a proof for Theorem 1.

**Lemma 2.** *[1] Let $t, u > 0$, $\boldsymbol{v}$ be an $m$-dimensional standard normal random vector, and let $F : \mathbb{R}^m \to \mathbb{R}$ be a function such that $\|F(\boldsymbol{x}) - F(\boldsymbol{y})\| \le \|\boldsymbol{x}-\boldsymbol{y}\|$ for all $\boldsymbol{x}$, $\boldsymbol{y} \in \mathbb{R}^m$. Then*

$$\mathbb{P}\left(F(\boldsymbol{v}) \ge \mathbb{E}F(\boldsymbol{v}) + u\right) \le \exp\left(-tu + \frac{1}{2}\left(\frac{\pi t}{2}\right)^2\right). \tag{25}$$

Taking $t = \frac{4}{\pi^2}$ in Lemma 2, we obtain

**Lemma 3.** *Let $u > 0$, $\boldsymbol{v}$ be an $m$-dimensional standard normal random vector, and let $F : \mathbb{R}^m \to \mathbb{R}$ be a function such that $\|F(\boldsymbol{x}) - F(\boldsymbol{y})\| \le \|\boldsymbol{x}-\boldsymbol{y}\|$ for all $\boldsymbol{x}$, $\boldsymbol{y} \in \mathbb{R}^m$. Then*

$$\mathbb{P}\left(F(\boldsymbol{v}) \ge \mathbb{E}F(\boldsymbol{v}) + u\right) \le \exp\left(-\frac{2}{\pi^2}u^2\right). \tag{26}$$

**Lemma 4.** *Let $\boldsymbol{v}$ be an $m$-dimensional standard normal random vector. Let $1 \leq p \leq \infty$. Let $0 < u < \mathbb{E}\|\boldsymbol{v}\|_{\ell_p}$. Let $\boldsymbol{T} \in \mathbb{R}^{m\times m}$ be such that $\|\boldsymbol{T}\boldsymbol{x}\|_{\ell_p} \leq \|\boldsymbol{x}\|_{\ell_p}$ for all $\boldsymbol{x} \in \mathbb{R}^m$. Then*

$$\mathbb{P}\left(\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} \geq \frac{\mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} + u}{\mathbb{E}\|\boldsymbol{v}\|_{\ell_p} - u}\|\boldsymbol{v}\|_{\ell_p}\right) \leq 2\exp\left(-\frac{2}{\pi^2}u^2\right).$$

*Proof.* By Lemma 3,

$$\mathbb{P}(\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} \geq \mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} + u) \leq e^{-\frac{2}{\pi^2}u^2}$$

and

$$\mathbb{P}(-\|\boldsymbol{v}\|_{\ell_p} \geq -\mathbb{E}\|\boldsymbol{v}\|_{\ell_p} + u) \leq e^{-\frac{2}{\pi^2}u^2}.$$

The second inequality gives

$$\mathbb{P}(\|\boldsymbol{v}\|_{\ell_p} \leq \mathbb{E}\|\boldsymbol{v}\|_{\ell_p} - u) \leq e^{-\frac{2}{\pi^2}u^2}.$$

Therefore,

$$\begin{aligned}&\mathbb{P}\left(\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} \geq \frac{\mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} + u}{\mathbb{E}\|\boldsymbol{v}\|_{\ell_p} - u}\|\boldsymbol{v}\|_{\ell_p}\right)\\ \leq&\mathbb{P}(\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} \geq \mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} + u) + \mathbb{P}(\|\boldsymbol{v}\|_{\ell_p} \leq \mathbb{E}\|\boldsymbol{v}\|_{\ell_p} - u) \leq 2e^{-\frac{2}{\pi^2}u^2}.\end{aligned}$$

□

**Lemma 5.** *Let $1 \leq p \leq 2$. Let $\boldsymbol{T} \in \mathbb{R}^{m\times m}$. Let $\boldsymbol{v}$ be an $m$-dimensional standard normal random vector. Then*

$$\mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} \leq m^{\frac{1}{p}-\frac{1}{2}}(\mathrm{Trace}\,\boldsymbol{T}^*\boldsymbol{T})^{\frac{1}{2}}\left(\mathbb{E}|\boldsymbol{v}_1|^p\right)^{\frac{1}{p}},$$

*where $\boldsymbol{v}_1$ is the first coordinate of $\boldsymbol{v}$.*

*Proof.* We write $\boldsymbol{T} = (\boldsymbol{T}_{i,j})_{1\leq i,j\leq n}$. Then

$$\begin{aligned}\mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_p} &= \mathbb{E}\left(\sum_{i=1}^n\left|\sum_{j=1}^n \boldsymbol{T}_{i,j}\boldsymbol{v}_j\right|^p\right)^{\frac{1}{p}}\\ &\leq \left(\sum_{i=1}^n\mathbb{E}\left|\sum_{j=1}^n \boldsymbol{T}_{i,j}\boldsymbol{v}_j\right|^p\right)^{\frac{1}{p}}\\ &= \left(\sum_{i=1}^n\left(\sum_{j=1}^n \boldsymbol{T}_{i,j}^2\right)^{\frac{p}{2}}\mathbb{E}|\boldsymbol{v}_1|^p\right)^{\frac{1}{p}}\\ &\leq \left(n^{1-\frac{p}{2}}\left(\sum_{1\leq i,j\leq n} \boldsymbol{T}_{i,j}^2\right)^{\frac{p}{2}}\mathbb{E}|\boldsymbol{v}_1|^p\right)^{\frac{1}{p}}\\ &= n^{\frac{1}{p}-\frac{1}{2}}\left(\mathrm{Trace}\ \boldsymbol{T}^*\boldsymbol{T}\right)^{\frac{1}{2}}\left(\mathbb{E}|\boldsymbol{v}_1|^p\right)^{\frac{1}{p}},\end{aligned}$$

where the second equality follows from the assumption that $\boldsymbol{v}$ is an $m$-dimensional standard normal random vector. □

**Lemma 6.** *Let $\boldsymbol{v}$ be an $m$-dimensional standard normal random vector. Then*

$$\mathbb{E}\|\boldsymbol{v}\|_{\ell_2} \geq \sqrt{m} - \pi.$$

*Proof.* By Lemma 3,

$$\mathbb{P}(\|\boldsymbol{v}\|_{\ell_2} \geq \mathbb{E}\|\boldsymbol{v}\|_{\ell_2} + u) \leq e^{-\frac{2}{\pi^2}u^2}$$

and

$$\mathbb{P}(-\|\boldsymbol{v}\|_{\ell_2} \geq -\mathbb{E}\|\boldsymbol{v}\|_{\ell_2} + u) \leq e^{-\frac{2}{\pi^2}u^2}.$$

Thus,

$$\mathbb{P}(|\|\boldsymbol{v}\|_{\ell_2} - \mathbb{E}\|\boldsymbol{v}\|_{\ell_2}| \geq u) \leq 2e^{-\frac{2}{\pi^2}u^2}.$$

Consider the random variable $W = \|\boldsymbol{v}\|_{\ell_2}$. We have

$$\mathbb{E}|W - \mathbb{E}W|^2 = \int_0^\infty \mathbb{P}(|W - \mathbb{E}W| \geq \sqrt{u})\, du \leq \int_0^\infty 2e^{-\frac{2}{\pi^2}u}\, du = \pi^2.$$

Since $\mathbb{E}|W - \mathbb{E}W|^2 = \mathbb{E}W^2 - (\mathbb{E}W)^2$, we have

$$\mathbb{E}W \geq (\mathbb{E}W^2)^{\frac{1}{2}} - (\mathbb{E}|W - \mathbb{E}W|^2)^{\frac{1}{2}} \geq \sqrt{m} - \pi.$$

□

**Lemma 7.** *Let $0 < \epsilon < 1 - \frac{\pi}{\sqrt{m}}$. Let $\sigma > 0$. Let*

$$\beta = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{1 + 2\sigma - \sigma z_i - \sigma \overline{z_i}},$$

*where $z_1, \ldots, z_m$ are the $m$ roots of unity. Let $\boldsymbol{B}$ be the circular shift operator on $\mathbb{R}^m$. Let $\boldsymbol{v}$ be an $m$-dimensional standard normal random vector. Then*

$$\mathbb{P}\left(\|((1+2\sigma)\boldsymbol{I} - \sigma\boldsymbol{B} - \sigma\boldsymbol{B}^*)^{-1/2}\boldsymbol{v}\|_{\ell_2} \geq \frac{\sqrt{\beta} + \epsilon}{1 - \frac{\pi}{\sqrt{m}} - \epsilon}\|\boldsymbol{v}\|_{\ell_2}\right) \leq 2e^{-\frac{2}{\pi^2}m\epsilon^2}.$$

*Proof.* Let $\boldsymbol{T} = ((1+2\sigma)\boldsymbol{I} - \sigma\boldsymbol{B} - \sigma\boldsymbol{B}^*)^{-1/2}$. Taking $u = \sqrt{m}\epsilon$ in Lemma 4, we have

$$\mathbb{P}\left(\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_2} \geq \frac{\mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_2} + \sqrt{m}\epsilon}{\mathbb{E}\|\boldsymbol{v}\|_{\ell_2} - \sqrt{m}\epsilon}\|\boldsymbol{v}\|_{l^2}\right) \leq 2e^{-\frac{2}{\pi^2}m\epsilon^2}.$$

By Lemma 5, $\mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_2} \leq (\mathrm{Trace}\,\boldsymbol{T}^*\boldsymbol{T})^{\frac{1}{2}}$. we have $\mathrm{Trace}\,\boldsymbol{T}^*\boldsymbol{T} = m\beta$. It is easy to show that $\mathrm{Trace}\,\boldsymbol{T}^*\boldsymbol{T}) = m\beta$ So $\mathbb{E}\|\boldsymbol{T}\boldsymbol{v}\|_{\ell_2} \leq \sqrt{m\beta}$. Also by Lemma 6, $\mathbb{E}\|\boldsymbol{v}\|_{\ell_2} \geq \sqrt{m} - \pi$. Therefore,

$$\mathbb{P}\left(\|((1+2\sigma)\boldsymbol{I} - \sigma\boldsymbol{B} - \sigma\boldsymbol{B}^*)^{-1}\boldsymbol{v}\|_{\ell_2} \geq \frac{\sqrt{\beta} + \epsilon}{1 - \frac{\pi}{\sqrt{m}} - \epsilon}\|\boldsymbol{v}\|_{\ell_2}\right) \leq 2e^{-\frac{2}{\pi^2}m\epsilon^2}.$$

□

*Proof of Theorem 1.* Theorem 1 follows from Lemma 7 by substituting $\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_{\ell_2}}$ and using homogeneity and direct calculations. □

## 9.2 Proof of Theorem 2

In this part, we will give a proof for Theorem 2.

**Lemma 8** ([6])**.** *Let $\prec_w$ denotes weak majorization. Denote eigenvalues of Hermitian matrix $\boldsymbol{X}$, by $\lambda_1(\boldsymbol{X}) \geq \ldots \geq \lambda_m(\boldsymbol{X})$. For every two Hermitian positive definite matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we have*

$$(\lambda_1(\boldsymbol{AB}), \cdots, \lambda_m(\boldsymbol{AB})) \prec_w (\lambda_1(\boldsymbol{A})\lambda_1(\boldsymbol{B}), \cdots, \lambda_m(\boldsymbol{A})\lambda_m(\boldsymbol{B})).$$

*In particular,*

$$\sum_{j=1}^{m} \lambda_j(\boldsymbol{AB}) \leq \sum_{j=1}^{m} \lambda_j(\boldsymbol{A})\lambda_j(\boldsymbol{B}).$$

*proof of Theorem 2.* Let $\lambda_1 \geq \ldots \geq \lambda_m$ denote the eigenvalues of $\Sigma$. The eigenvalues of $(A_\sigma^n)^{-2}$ are given by $\{[1 + 4^n\sigma\sin^{2n}(\pi j/m)]^{-2}\}_{j=0}^{j=m-1}$, which we denote by $1 = \alpha_1 \geq \ldots \geq \alpha_m \geq (1 + 4^n\sigma)^{-2}$. We have

$$\sum_{j=1}^{m} \mathrm{Var}[\boldsymbol{n}_j] = \mathrm{trace}(\Sigma) = \sum_{j=1}^{m} \lambda_j. \tag{27}$$

On the other hand we also have

$$\sum_{j=1}^{m} \mathrm{Var}[(\boldsymbol{A}_\sigma^n)^{-1}\boldsymbol{n}_j] = \mathrm{trace}((\boldsymbol{A}_\sigma^n)^{-1}\Sigma(\boldsymbol{A}_\sigma^n)^{-1}) = \mathrm{trace}((\boldsymbol{A}_\sigma^n)^{-2}\Sigma) \leq \sum_{j=1}^{m} \alpha_j\lambda_j, \tag{28}$$

where the last inequality is by lemma 8. Now,

$$\begin{aligned}\sum_{j=1}^{m}\lambda_j - \sum_{j=1}^{m}\alpha_j\lambda_j &= \sum_{j=1}^{m}(1-\alpha_j)\lambda_j \\ &\geq \lambda_m(m - \sum_{j=1}^{m}\alpha_j) \\ &= \frac{\lambda_1}{\kappa}(m - \sum_{j=1}^{m}\alpha_j) \\ &\geq \frac{\sum_{j=1}^{m}\lambda_j}{m\kappa}(m - \sum_{j=1}^{m}\alpha_j)\end{aligned}$$

Rearranging and simplifying above implies that

$$\sum_{j=1}^{m}\alpha_j\lambda_j \leq (\sum_{j=1}^{m}\lambda_j)(1 - \frac{1}{\kappa} + \frac{\sum_{j=1}^{m}\alpha_j}{m\kappa}).$$

Substituting Eq. (27) and Eq. (28) in the above inequality, yields Eq. (12). □

## 9.3 Proof of Lemma 1

To proof Lemma 1, we first introduce the following lemma.

**Lemma 9.** *For $0 \leq \theta \leq 2\pi$, suppose*

$$F(\theta) = \frac{1}{1 + 2\sigma(1-\cos(\theta))},$$

*has the discrete-time Fourier transform of series $f[k]$. Then, for integer $k$,*

$$f[k] = \frac{\alpha^{|k|}}{\sqrt{4\sigma+1}}$$

*where*

$$\alpha = \frac{2\sigma + 1 - \sqrt{4\sigma+1}}{2\sigma}$$

*Proof.* By definition,

$$f[k] = \frac{1}{2\pi}\int_0^{2\pi} F(\theta)e^{ik\theta}\,d\theta = \frac{1}{2\pi}\int_0^{2\pi}\frac{e^{ik\theta}}{1+2\sigma(1-\cos(\theta))}\,d\theta. \tag{29}$$

Computing Eq. (29) using Residue Theorem is a well-known technique in complex analysis. First, note that because $F(\theta)$ is real valued, $f[k] = f[-k]$; therefore, it suffices to compute Eq. (29) for nonnegative $k$. Set $z = e^{i\theta}$. Observe that $\cos(\theta) = 0.5(z + 1/z)$ and $dz = izd\theta$. Substituting in Eq. (29) and simplifying yields that

$$f[k] = \frac{-1}{2\pi i\sigma}\oint \frac{z^k}{(z-\alpha_-)(z-\alpha_+)}\,dz, \tag{30}$$

where the integral is taken around the unit circle, and $\alpha_\pm = \frac{2\sigma+1\pm\sqrt{4\sigma+1}}{2\sigma}$ are the roots of quadratic $-\sigma z^2 + (2\sigma+1)z - \sigma$. Note that $\alpha_-$ lies within the unit circle; whereas, $\alpha_+$ lies outside of the unit circle. Therefore, because $k$ is nonnegative, $\alpha_-$ is the only singularity of the integrand in Eq. (30) within the unit circle. A straightforward application of the Residue Theorem yields that

$$f[k] = \frac{-\alpha_-^k}{\sigma(\alpha_- - \alpha_+)} = \frac{\alpha^k}{\sqrt{4\sigma+1}}.$$

This completes the proof. □

Next, we give a proof for Lemma 1.

*Proof of Lemma 1.* First observe that we can re-write the left hand side of Eq. (9) as

$$\frac{1}{m}\sum_{j=0}^{m-1}\frac{1}{1+2\sigma(1-\cos(\frac{2\pi j}{m}))}. \tag{31}$$

It remains to show that the above summation is equal to the right hand side of Eq. (9). This follows by lemmas 9 and standard sampling results in Fourier analysis (i.e. sampling $\theta$ at points $\{2\pi j/m\}_{j=0}^{m-1}$). Nevertheless, we provide the details here for completeness: Observe that that the inverse discrete-time Fourier transform of

$$G(\theta) = \sum_{j=0}^{m-1}\delta(\theta - \frac{2\pi j}{m}).$$

is given by

$$g[k] = \begin{cases} m/2\pi & \text{if } k \text{ divides } m, \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, let

$$F(\theta) = \frac{1}{1+2\sigma(1-\cos(\theta))},$$

and use $f[k]$ to denote its inverse discrete-time Fourier transform. Now,

$$\begin{aligned}
\frac{1}{m}\sum_{j=0}^{m-1}\frac{1}{1+2\sigma(1-\cos(\frac{2\pi j}{m}))} &= \frac{1}{m}\int_0^{2\pi} F(\theta)G(\theta) \\
&= \frac{2\pi}{m}\,\mathrm{DTFT}^{-1}[F\cdot G][0] \\
&= \frac{2\pi}{m}(\mathrm{DTFT}^{-1}[F] * \mathrm{DTFT}^{-1}[G])[0] \\
&= \frac{2\pi}{m}\sum_{r=-\infty}^{\infty} f[-r]g[r] \\
&= \frac{2\pi}{m}\sum_{\ell=-\infty}^{\infty} f[-\ell m]\frac{m}{2\pi} \\
&= \sum_{\ell=-\infty}^{\infty} f[-\ell m].
\end{aligned}$$

The proof is completed by substituting the result of lemma 9 in the above sum and simplifying. □

## Acknowledgments

This material is based on research sponsored by the Air Force Research Laboratory under grant numbers FA9550-18-0167 and MURI FA9550-18-1-0502, the Office of Naval Research under grant number N00014-18-1-2527, the U.S. Department of Energy under grant number DOE SC0013838, and by the National Science Foundation under grant number DMS-1554564, (STROBE). We would like to thank Jialin Liu and professors Pratik Chaudhari, Adam Oberman and Ming Yan for stimulating discussions.

## References

[1] 254a, notes 1: Concentration of measure. `https://terrytao.wordpress.com/2010/01/03/254a-notes-1-concentration-of-measure/`.

[2] M. Abadi, A. Agarwal, and et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[3] Z. Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18:1–51, 2018.

[4] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[5] D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[6] R. Bhatia. *Matrix Analysis*. Springer, 1997.

[7] L. Bottou. Stochastic gradient descent tricks. *Neural Networks, Tricks of the Trade, Reloaded*, 7700, 2012.

[8] L. Bottou, E. F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[10] P. Chaudhari, A. Oberman, S. Osher, S. Soatto, and C. Guillame. Deep relaxation: partial differential equations for optimizing deep neural networks. *arXiv preprint arXiv:1704.04932*, 2017.

[11] A. Defazio and F. Bach. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, 2014.

[12] T. Dozat. Incorporating nesterov momentum into adam. In *4th International Conference on Learning Representation Workshop (ICLR 2016)*, 2016.

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

[14] L.C. Evans. Partial differential equations. 2010.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[16] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *33rd International COnference on Machine Learning (ICML 2016)*, 2016.

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

[18] S. Jastrzebski, Z. Kenton, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Dnn's sharpest directions along the sgd trajectory. *arXiv preprint arXiv:1807.05031*, 2018.

[19] R. Johoson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 2013.

[20] M. Jung, G. Chung, G. Sundaramoorthi, L. Vese, and A. Yuille. Sobolev gradients and joint variational image segmentation, denoising, and deblurring. In *Computational Imaging VII*, volume 7246, page 72460I. International Society for Optics and Photonics, 2009.

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

[22] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 81:2278–2324, 1998.

[24] L. Lei, C. Ju, J. Chen, and M. Jordan. Nonconvex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, 2017.

[25] F. Li and et al. Cs231n: Convolutional neural networks for visual recognition. 2018.

[26] H. Li, Z. Xu, G. Taylor, and T. Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[27] S. Chintala M. Arjovsky and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[28] S. Mandt, M. Hoffman, and D. Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18:1–35, 2017.

[29] Mnih and et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[31] Y. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl. akad. nauk Sssr*, 269:543–547, 1983.

[32] Y. Nesterov. Introductory lectures on convex programming volume i: Basic course. *Lecture Notes*, 1998.

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[34] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks : The Official Journal of the International Neural Network Society*, 12(1):145–151, 1999.

[35] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[36] S. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representation (ICLR 2018)*, 2018.

[37] H. Robinds and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[38] J Schmidhuber. Deep learning in neural networks: An overview. *arXiv preprint arXiv:1404.7828*, 2014.

[39] A. Senior, G. Heigold, M. Ranzato, and K. Yang. An empirical study of learning rates in deep neural networks for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[40] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *30th International Conference on Machine Learning (ICML 2013)*, 2013.

[41] A. Shapiro and Y. Wardi. Convergence analysis of gradient descent stochastic algorithms. *Journal of Optimization Theory and Applications*, 91(2):439–454, 1996.

[42] D. Silver and et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[43] R. Sutton. Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proc. 8th Annual Conf. Cognitive Science Society*, 1986.

[44] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[45] B. Wang, Q. Gu, M. Boedihardjo, F. Barekat, and S. Osher. Privacy-preserving erm by laplacian smoothing stochastic gradient descent. *UCLA Computational and Applied Mathematics Reports*, 19-24, 2019.

[46] M. Welling and Y. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *28th International Conference on Machine Learning (ICML 2011)*, 2011.

[47] Y. Wu and K. He. Group normalization. In *European Conference on Computer Vision*, 2018.

[48] M. Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.