
Extrapolation for Large-batch Training in Deep Learning

Tao Lin^{*1} Lingjing Kong^{*1} Sebastian U. Stich¹ Martin Jaggi¹

Abstract

Deep learning networks are typically trained by Stochastic Gradient Descent (SGD) methods that iteratively improve the model parameters by estimating a gradient on a very small fraction of the training data. A major roadblock faced when increasing the batch size to a substantial fraction of the training data for reducing training time is the persistent degradation in performance (generalization gap). To address this issue, recent work propose to add small perturbations to the model parameters when computing the stochastic gradients and report improved generalization performance due to smoothing effects. However, this approach is poorly understood; it requires often model-specific noise and fine-tuning.

To alleviate these drawbacks, we propose to use instead computationally efficient extrapolation (extragradient) to stabilize the optimization trajectory while still benefiting from smoothing to avoid sharp minima. This principled approach is well grounded from an optimization perspective and we show that a host of variations can be covered in a unified framework that we propose. We prove the convergence of this novel scheme and rigorously evaluate its empirical performance on ResNet, LSTM, and Transformer. We demonstrate that in a variety of experiments the scheme allows scaling to much larger batch sizes than before whilst reaching or surpassing SOTA accuracy.

1. Introduction

The workhorse training algorithm for most machine learning applications—including deep learning—is Stochastic Gradient Descent (SGD). Recently, data parallelism has emerged in deep learning, where large-batch (Goyal et al.,

2017) is used to reduce the gradient computation rounds so as to accelerate the training. However, in practice, these large-batch variants suffer from severe issues of test quality loss (Shallue et al., 2018; McCandlish et al., 2018; Golmant et al., 2018), which voids gained computational advantage. While still not completely understood, recent research has linked part of this loss of efficiency to the existence of sharp minima. In contrast, landscapes with flat minima have in empirical studies shown generalization benefits (Keskar et al., 2016; Yao et al., 2018; Lin et al., 2020), though this topic is still actively debated (Dinh et al., 2017).

Another line of research tries to understand general deep learning training from an optimization perspective, in terms of the optimization trajectory in the loss surface (Neyshabur, 2017; Jastrzebski et al., 2020). Golatkar et al. (2019) empirically show that regularization techniques only affect early learning dynamics (initial optimization phase) but matter little in the final phase of training (converging to a local minimum), similar to the critical initial learning phase described in (Achille et al., 2019) and the break-even point analysis on the entire optimization trajectory of Jastrzebski et al. (2020).

These new insights are consistent with empirically developed techniques for the SOTA large-batch training. For example, gradual learning rate warmup for the first few epochs (Goyal et al., 2017; You et al., 2019) is often used; and in local SGD (Lin et al., 2020) for better generalization, stochastic noise injection is only applied *after* the first phase of training (post-local SGD).

These discussions on optimization and generalization motivate us to answer the following questions when using or developing large-batch techniques for better training: *Does the proposed technique improve (initial) optimization, or help to converge to a better local minimum, or both? How and when should we apply the technique?*

In this paper, we first revisit the classical smoothing idea (which was recently attributed to avoiding sharp minima in deep learning (Wen et al., 2018; Haruki et al., 2019)) from optimization perspective. We then propose a computational efficient local extragradient method as a way of smoothing for distributed large-batch training, referred to as EXTRAP-SGD; we further extend it to a general framework (extrapolated SGD) for distributed training. We thoroughly evaluate

^{*}Equal contribution ¹EPFL, Lausanne, Switzerland. Correspondence to: Tao Lin <tao.lin@epfl.ch>.

our method on diverse tasks to understand how and when it improves the training in practice. Our empirical results justify the benefits of EXTRAP-SGD, and explain the effects of smoothing ill-conditioned loss landscapes as to exhibit more well-behaved regions, which contain multiple solutions of good generalization properties (Garipov et al., 2018). We show the importance of using EXTRAP-SGD in the critical initial optimization phase, for the later convergence to better local minima; the conjecture is verified by the combination with post-local SGD, which achieves SOTA large-batch training. Our main contributions can be summarized as follows:

- We propose EXTRAP-SGD and extend it to a unified framework (extrapolated SGD) for distributed large-batch training. Extensive empirical results on three benchmarking tasks justify the effects of accelerated optimization and better generalization.
- We provide convergence analysis for methods in the proposed framework, as well as the SOTA large batch training method (i.e. mini-batch SGD with Nesterov momentum). Our analysis explains the large batch optimization inefficiency (diminishing linear speedup) observed in previous empirical work.

2. Related Work

Large-batch training. The test performance degradation (often denoted as *generalization gap*) caused by large batch training has recently drawn significant attention (Keskar et al., 2016; Hoffer et al., 2017; Shallue et al., 2018; Masters & Luschi, 2018). Hoffer et al. (2017) argue that the generalization gap in some cases can be closed by increasing training iterations and adjusting the learning rate proportional to the square root of the batch size. Goyal et al. (2017) argue the poor test performance is due to the optimization issue; they try to bridge the generalization gap with the heuristics of linear scale-up of the learning rate during training or during a warmup phase. You et al. (2017) propose Layer-wise Adaptive Rate Scaling (LARS) for better optimization and scaling to larger mini-batch sizes; but the generalization gap does not vanish. Lin et al. (2020) further propose post-local SGD on top of these optimization techniques to inject stochastic noise (to mimic the training dynamics of small batch SGD) during the later training phase.

In addition to the techniques developed for improving optimization and generalization, the optimization ineffectiveness (in terms of required training steps-to-target performance) of large-batch training has been observed. Shallue et al. (2018); McCandlish et al. (2018); Golmant et al. (2018) empirically demonstrate the existence of diminishing linear speedup region across different domains and architectures. Such a limit is also theoretically characterized in (Ma et al., 2017; Yin et al., 2017) for mini-batch SGD in the convex set-

ting.

Smoothing the “sharp minima”. Some research links generalization performance to flatness of minima. Entropy SGD (Chaudhari et al., 2017) proposes Langevin dynamics in the inner optimizer loop to smoothen out sharp valleys of the loss landscape. From the perspective of large-batch training, Wen et al. (2018) perform “sequential averaging” over models perturbed by isotropic noise, as a way to combat sharp minima. Haruki et al. (2019) claim that injecting different anisotropic stochastic noises on local workers can smoothen sharper minima. However, the claimed “sharper minima” is debatable (Dinh et al., 2017); it is also unclear whether the improved results are due to obtained flatter local minima or improved initial optimization brought by smoothing. We defer detailed discussion to Section 3.2 and 5.2.

Smoothing in classical optimization. Randomized smoothing has a long history in the optimization literature, see e.g. Nesterov (2011); Duchi et al. (2012); Scaman et al. (2018) which show that a faster convergence rates can be achieved by convolving non-smooth convex functions with Gaussian noise. In contrast to non-smooth convex functions, we focus on the smooth non-convex functions, motivated by deep neural networks.

Extragradient methods and optimization stability. Another useful building block from optimization is the extragradient method, which is a well-known technique to stabilize the training at each iteration by approximating the implicit update. The method was first introduced in (Korpelevich, 1976) and extended to many variants, e.g. mirror-prox (Nemirovski, 2004), Optimistic Mirror Descent (OMD) (Juditsky et al., 2011) (using past gradient information), extragradient method with Nesterov momentum (Diakonikolas & Orecchia, 2017). Recently its stochastic variants have found new applications in machine learning, e.g., Generative Adversarial Network (GAN) training (Daskalakis et al., 2017; Gidel et al., 2018; Chavdarova et al., 2019; Mishchenko et al., 2019), and low bit model training (Leng et al., 2017).

On the theoretical side, several papers analyze the convergence of stochastic variants of extragradient. Juditsky et al. (2011) study stochastic mirror-prox under restrictive assumptions. Xu et al. (2019) analyze stochastic extragradient in a more general non-convex setting and demonstrate tighter upper bounds than mini-batch SGD, when using mini-batch size $\mathcal{O}(1/\epsilon^2)$. Mishchenko et al. (2019) revisit and slightly extend the stochastic extragradient for better implicit update approximation. However, their work focuses on min-max GAN training and argues the stochastic extragradient method might not be better than SGD for traditional function minimizations tasks. Our work is the first that com-

bines the idea of Nesterov momentum and extragradient (from past information) for stochastic optimization in the setting of distributed training.

3. Optimization with Extrapolation

Problem Setting and Notation. We consider sum-structured optimization problems of the form $\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N f_i(\mathbf{x})$, where \mathbf{x} denotes the parameters of the model (neural networks in our case), and f_i denotes the loss function of the i -th (out of N) training data examples. To introduce our notation, we recall a standard update of mini-batch SGD at iteration t , computed on K devices:

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_t \left[\frac{1}{KB} \sum_{k=1}^K \sum_{i \in \mathcal{I}_t^k} \nabla f_i(\mathbf{x}_t) \right]. \quad (1)$$

Here \mathcal{I}_t^k denotes a subset of the training points selected on device k (typically selected uniformly at random) and we denote by $B := |\mathcal{I}_t^k|$ the local mini-batch size and by γ_t the step-size (learning rate).

3.1. Accelerated (Stochastic) Local Extragradient for Distributed Training

Motivated by the idea of randomized smoothing in the classic optimization literature (as for reducing the Lipschitz constant of the gradient), we here introduce the novel idea of using extragradient locally, as a way of smoothing loss surface, for efficient distributed large-batch training.

The original idea of extrapolation (or extragradient (Korpelevich, 1976)) was developed to stabilize optimization dynamics on saddle-point problems for a single worker, such as e.g. in GAN training (Gidel et al., 2018; Chavdarova et al., 2019). The idea is to compute the gradient at an extrapolated point, different from the current point from which the update will be performed:

$$\mathbf{x}_{t+\frac{1}{2}} = \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t), \quad \mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_{t+\frac{1}{2}}).$$

This step is intrinsically different from the well-known and widely used accelerated method (i.e. Nesterov momentum):

$$\begin{aligned} \mathbf{x}_{t+\frac{1}{2}} &= \mathbf{x}_t + u\mathbf{v}_t, \mathbf{v}_{t+1} = u\mathbf{v}_t - \gamma \nabla f(\mathbf{x}_{t+\frac{1}{2}}), \\ \mathbf{x}_{t+1} &= \mathbf{x}_{t+\frac{1}{2}} - \gamma \nabla f(\mathbf{x}_{t+\frac{1}{2}}), \end{aligned}$$

where here $1 > u \geq 0$ denotes the momentum parameter. The key difference lies in the lookahead step for the gradient computation of these two methods.

Considering different extrapolated local models (with Nesterov momentum) under the distributed training, EXTRAP-SGD combines the effects of randomized smoothing and the stabilized optimization dynamics through extrapolation. Our EXTRAP-SGD follows the idea of extrapolating

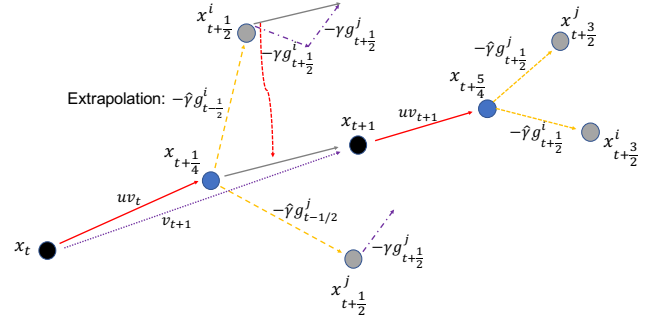


Figure 1. Illustrated EXTRAP-SGD ($K=2$).

from the past (currently only used for single worker training (Gidel et al., 2018; Daskalakis et al., 2017)), as a means to avoid additional cost for gradients used to form an extrapolation point. The EXTRAP-SGD method is detailed in Algorithm 1 and illustrated in Figure 1¹, where the previous local gradients are used for the extrapolation and the superscript k stresses that local models are different. Using the past local mini-batch gradients ($\nabla f_i(\mathbf{x}_{t-\frac{1}{2}}^k)$) for extrapolation (at time t) is a computationally efficient and allows the extrapolation scale $\hat{\gamma}$ to directly take on the learning rate used for small mini-batch training with size B , thus avoids the difficulty of hyper-parameter tuning. Note that setting $\hat{\gamma} = 0$ in Algorithm 1 (line 2) recovers the SOTA large batch training method, i.e., mini-batch SGD with Nesterov momentum.

To the best of our knowledge, it is the first time such an extragradient method is used locally with Nesterov Momentum under the framework of smoothing, for accelerated and smoothed distributed optimization.

Algorithm 1 EXTRAP-SGD

input: learning rate γ , inner learning rate $\hat{\gamma}$, momentum factor u , initial parameter \mathbf{x}_0 , initial moment vector $\mathbf{v}_0 = 0$, time step $t=0$, worker index k .

- 1: **while** \mathbf{x}_t not converged **do**
- 2: $\mathbf{x}_{t+\frac{1}{4}}^k = \mathbf{x}_t - \frac{\hat{\gamma}}{B} \sum_{i \in \mathcal{I}_t^k} \nabla f_i(\mathbf{x}_{t-\frac{1}{2}}^k) \triangleright$ extrapolation step
- 3: $\mathbf{x}_{t+\frac{1}{2}}^k = \mathbf{x}_{t+\frac{1}{4}}^k + u\mathbf{v}_t \triangleright$ Nesterov momentum
- 4: $\mathbf{v}_{t+1} = u\mathbf{v}_t - \frac{\gamma}{KB} \sum_{k,i \in \mathcal{I}_t^k} \nabla f_i(\mathbf{x}_{t+\frac{1}{2}}^k) \triangleright$ update buffer
- 5: $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1} \triangleright$ actual update
- 6: **end while**

output: \mathbf{x}_t .

3.2. Unified Extrapolation Framework

Our Algorithm 1 can be extended to a more general extrapolation framework for distributed training, by using diverse extrapolation choices ζ_t^k . It is achieved by replacing line 2 in Algorithm 1 by $\mathbf{x}_{t+\frac{1}{4}}^k = \mathbf{x}_t - \hat{\gamma}\zeta_t^k$. We denote our framework as *extrapolated SGD*, covering different choices of noise ζ_t^k

¹ we omit the extrapolation step in line 2 when $t=0$.

(e.g. Gaussian noise, uniform noise, and stochastic gradient noise) and EXTRAP-SGD (past mini-batch gradients). We detail some choices of ζ_t^k below and use them as our close baselines for EXTRAP-SGD:

- ζ_t^k as a form of isotropic noise. The noise can be sampled from e.g. an isotropic Gaussian or uniform distribution. Following the idea of Li et al. (2017), the strength of noises added to a filter can be linearly scaled by the l_2 norm of the filter, instead of fixing a constant perturbation strength over different layers. Formally, the scaled noise ζ_t^k of j -th filter at layer i on worker k follows $\|\mathbf{x}_{t,i,j}\| \cdot \hat{\zeta}_{t,i,j}^k / \|\hat{\zeta}_{t,i,j}^k\|$. A similar idea was proposed in SmoothOut (Wen et al., 2018), corresponding to letting $\zeta_t^k := \zeta_t$ in our framework.
- ζ_t^k as a form of anisotropic noise. Kleinberg et al. (2018) interpret sequential SGD updates as GD with stochastic gradient noise convolution over update steps. This motivates to use stochastic gradient noise for smoothing (similarly proposed in Haruki et al. (2019)), thus ζ_t^k can be chosen as:

$$\frac{1}{B} \sum_{i \in \mathcal{I}_t^k} \nabla f_i(\mathbf{x}_{t-\frac{1}{2}}^k) - \frac{1}{KB} \sum_k \sum_{i \in \mathcal{I}_t^k} \nabla f_i(\mathbf{x}_{t-\frac{1}{2}}^k).$$

The side effects of these noise extrapolation variants are the training setup sensitivity, causing the hyperparameter tuning difficulty and limited practical applications. For example, the isotropic noise requires to manually design the noise distribution for each model and dataset; the anisotropic noise distribution will be dynamically varied by different choices of the number of workers, the local mini-batch size, and the objective of the learning task (Zhang et al., 2019).

Despite the existence of variants (Wen et al., 2018; Haruki et al., 2019) and their reported empirical results², none of them has analyzed their convergence behaviors. In the next section, we provide rigorous convergence analysis for our algorithm for distributed training (illustrated in Algorithm 1, which also includes the SOTA practical training algorithm). In Section 5, we empirically evaluate all related methods to better understand the benefits of using extrapolation with smoothing for distributed large-batch training.

4. Theoretical Analysis of Nesterov Momentum and EXTRAP-SGD

We now turn to the theoretical convergence analysis, i.e. we derive an upper bound on the number of iterations to find an approximate solution with small gradient norm.

Following the convention in distributed stochastic optimization, We denote by f^* a lower bound on the values of $f(\mathbf{x})$

² The empirical results of Haruki et al. (2019) are not solid. Taking the results of CIFAR-10 for mini-batch size 8,192 into account and use three trials' experimental results for the same choice of ζ_t^k (e.g. layerwise uniform noise) as an example, our experimental results can reach reasonable test top-1 accuracy (at around 91), much better than their presented results (at around 63).

and use the following assumptions:

Assumption 1 (Unbiased Stochastic Gradients). $\forall i \in [N], t \in [T]$, it holds $\mathbb{E}[\nabla f_i(\mathbf{x}_t)] = \nabla f(\mathbf{x}_t)$.

Assumption 2 (Bounded Gradient Variance). $\exists \sigma^2 > 0, \forall i \in [N], t \in [T]$, s.t. $\mathbb{E}[\|\nabla f_i(\mathbf{x}_t) - \nabla f(\mathbf{x}_t)\|^2] \leq \sigma^2$.

Here σ^2 quantifies the variance of stochastic gradients at each local worker and we assume workers access IID training dataset (e.g., data center setting).

Assumption 3 (Lipschitz Gradient). $\exists L > 0$, s.t. $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, i \in [N]$, the objective function $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ satisfies the following condition $\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\|$.

4.1. Analysis for Mini-batch SGD (with Nesterov Momentum)

In this section we first recall the known convergence guarantees for mini-batch SGD (without momentum) on non-convex functions for later reference, and then derive new guarantees for mini-batch SGD with Nesterov momentum.

Theorem 4.1 (Convergence of stochastic distributed mini-batch SGD for non-convex functions (Ghadimi & Lan, 2016)). *Under Assumptions 1–3, after T mini-batch gradient updates, each using KB samples, the mini-batch SGD returns an iterate \mathbf{x} satisfying*

$$\mathbb{E}[\|\nabla f(\mathbf{x})\|^2] \leq \mathcal{O}\left(\frac{L(f(\mathbf{x}_0) - f^*)}{T} + \frac{\sigma\sqrt{L(f(\mathbf{x}_0) - f^*)}}{\sqrt{KBT}}\right).$$

The second term in the rate is asymptotically dominant as long as $KB = \mathcal{O}\left(\frac{\sigma^2 T}{L(f(\mathbf{x}_0) - f^*)}\right)$. In this regime, increasing the mini-batch size gives a linear speedup, as $T = \mathcal{O}\left(\frac{\sigma^2 L(f(\mathbf{x}) - f^*)}{KB\epsilon^2}\right)$ decreases in KB , as similarly pointed out by Wang & Srebro (2017). However, when we increase the mini-batch size beyond this critical point, the first term dominates the rate and increasing the mini-batch size further will have less impact on the convergence. This phenomenon has also been empirically verified in deep learning applications (Shallue et al., 2018).

In practice, mini-batch SGD with Nesterov momentum (Nesterov, 1983) is the state-of-the-art deep learning training scheme. However, previous theoretical analysis normally relies on the strong assumption of the bounded mini-batch gradients (Yan et al., 2018). Here we provide a better convergence analysis without such an assumption for distributed mini-batch SGD with Nesterov momentum following closely Yu et al. (2019). The convergence rate is detailed in Theorem 4.2, and for the proof details we refer to Section A of Appendix I.

Theorem 4.2 (Convergence of mini-batch SGD with Nesterov momentum for non-convex functions). *Under Assumption 1–3, for mini-batch SGD with Nesterov momentum, i.e., $\mathbf{x}_{t+\frac{1}{2}} = \mathbf{x}_t + u\mathbf{v}_t$, $\mathbf{v}_{t+1} = u\mathbf{v}_t -$*

$\frac{\gamma}{KB} \sum_{k=1}^K \sum_{i \in \mathcal{I}_t^k} \nabla f_i(\mathbf{x}_{t+\frac{1}{2}})$, $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}$, we can show that for optimally tuned stepsize (cf. Lemma A.4) $\gamma \leq \frac{2(1-u)^2}{L(u^3+1)}$, it holds

$$\mathbb{E} \left[\|\nabla f(\mathbf{x})\|^2 \right] = \mathcal{O} \left(\frac{Lr_0(u^3+1)}{T(1-u)} + \sqrt{\frac{2Lr_0\sigma^2}{KBT(1-u)}} \right),$$

where here \mathbf{x} denotes a uniformly at random selected $\mathbf{x}_{t+\frac{1}{2}}$ iterate, i.e. $\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_{t+\frac{1}{2}})\|^2$, and $r_0 := f(\mathbf{x}_0) - f^*$.

The second term is asymptotically dominant as long as $KB = \mathcal{O} \left(\frac{(1-u)}{(u^3+1)^2} \frac{\sigma^2 T}{L(f(\mathbf{x}_0) - f^*)} \right)$ and for small K we can achieve the linear speedup where $T = \mathcal{O} \left(\frac{L\sigma^2(f(\mathbf{x}_0) - f^*)}{(1-u)KB\epsilon^2} \right)$.

Remark 4.3. Yu et al. (2019) argue that a linear speedup $\mathcal{O}(1/\sqrt{KT})$ for SGD with the local Nesterov momentum can be achieved. However, this claim is only valid for large T and stepsize $\gamma = \sqrt{K/T}$, cf. (Yu et al., 2019, Cor. 1). We here tune the stepsize differently and show a tighter bound that holds for all T , providing a better critical mini-batch size analysis (diminishing linear speedup in terms of optimization) for mini-batch SGD with Nesterov momentum (Shallue et al., 2018).

4.2. Convergence of EXTRAP-SGD

In this subsection, we show the convergence analysis for our novel EXTRAP-SGD for non-convex functions. We also include the analysis for other noise variants of extrapolated SGD, which explains their potential limitations. The proof details can be found in the Section B of Appendix I.

Theorem 4.4 (Convergence of EXTRAP-SGD for non-convex functions). Under Assumption 1–3, and by defining $\bar{\mathbf{x}}_{t+\frac{1}{2}} := \frac{1}{K} \sum_{k=1}^K \mathbf{x}_{t+\frac{1}{2}}^k$, it holds for $\hat{\gamma} \leq \frac{u^2}{(1-u)^2} \gamma$ and $\gamma \leq \frac{(1-u)^2}{L(1+3u+u^3)}$:

$$\begin{aligned} & \mathbb{E} \frac{1}{T} \sum_{t=0}^{T-1} \left\| \nabla f(\bar{\mathbf{x}}_{t+\frac{1}{2}}) \right\|^2 \\ & \leq \frac{2(1-u)}{\gamma T} \mathbb{E} [f(\bar{\mathbf{x}}_0) - f^*] + \left(\frac{4\hat{\gamma}^2 L^2}{B} + \frac{\gamma L(1+3u)}{(1-u)^2 BK} \right) \sigma^2. \end{aligned}$$

Remark 4.5. Using past local gradients for extrapolation in EXTRAP-SGD allows us to directly set $\hat{\gamma} \approx \frac{\gamma}{K}$ for EXTRAP-SGD, where the constraint of $\hat{\gamma} \leq \frac{u^2}{(1-u)^2} \gamma$ is normally satisfied.

Corollary 4.6. Considering Theorem 4.4 and tuning the stepsize as in Lemma A.4, with $\gamma \leq \frac{(1-u)^2}{L(1+3u+u^3)}$ and $\hat{\gamma} \leq \frac{\gamma}{K}$, and $r_0 := f(\mathbf{x}_0) - f^*$, we can rewrite the convergence rate of Theorem 4.4 as

$$\begin{aligned} & \mathbb{E} \left[\frac{1}{T} \sum_{t=0}^{T-1} \left\| \nabla f(\bar{\mathbf{x}}_{t+\frac{1}{2}}) \right\|^2 \right] \\ & = \mathcal{O} \left(4(u^3 + 3u + 1) \frac{Lr_0}{T(1-u)} + 2\sqrt{\frac{(19u+1)}{(u^3+3u+1)} \frac{2Lr_0\sigma^2}{KBT(1-u)}} \right). \end{aligned}$$

The second term is asymptotically dominant as long as $KB = \mathcal{O} \left(\frac{(19u+1)(1-u)}{(u^3+3u+1)^3} \frac{\sigma^2 T}{L(f(\mathbf{x}_0) - f^*)} \right)$ and for small K we can achieve the linear speedup where $T = \mathcal{O} \left(\frac{L\sigma^2(f(\mathbf{x}_0) - f^*)}{KB\epsilon^2} \frac{19u+1}{(u^3+3u+1)(1-u)} \right)$,

Remark 4.7. By setting $u = 0$, we recover the same rates for EXTRAP-SGD in non-convex cases (Theorem 4.2 and Corollary 4.6) as for standard mini-batch SGD (Theorem 4.1) but we cannot show an actual speedup over mini-batch SGD by setting $u > 0$. However, this might not necessarily be a limitation of our approach as to the best of our knowledge, there exist so far no theoretical results for stochastic momentum methods that can show a speedup over mini-batch SGD.

The analysis below extends the proof of EXTRAP-SGD to the other cases of our extrapolated SGD framework.

Theorem 4.8. Under the extrapolation framework, IID random noise ζ_t^k (instead of the past local mini-batch gradients in Algorithm 1) is used for the local extrapolation, where $\mathbb{E}[\zeta_t^k] = 0$ and $\mathbb{E}[\|\zeta_t^k\|^2] \leq \hat{\sigma}^2$. Under Assumption 1–3, it holds for $\gamma \leq \frac{(1-u)^2}{L(1+u+u^3)}$:

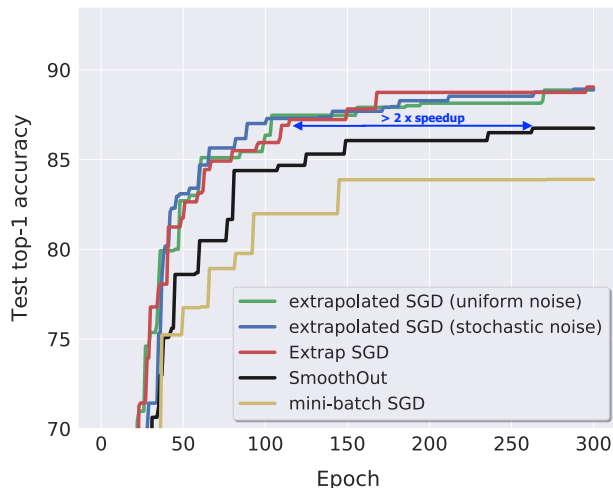
$$\begin{aligned} & \mathbb{E} \frac{1}{T} \sum_{t=0}^{T-1} \left\| \nabla f(\bar{\mathbf{x}}_{t+\frac{1}{2}}) \right\|^2 \leq \frac{2(1-u)}{\gamma T} \mathbb{E} [f(\bar{\mathbf{y}}_0) - f(\bar{\mathbf{y}}_T)] \\ & + \frac{\gamma L(1+u)}{(1-u)^2 BK} \sigma^2 + (L^2 + \frac{(1-u)^2 L}{\gamma u^3 K}) 2\hat{\gamma}^2 T \hat{\sigma}^2. \end{aligned}$$

Remark 4.9. The choice of using random noise for extrapolation in Theorem 4.8 requires the manual introduction of the noise distribution ($\hat{\sigma}^2$) for each problem setup. The dependence on the unknown relationship between $\hat{\sigma}^2$ and σ^2 results in the difficulty of providing a concise convergence analysis (e.g. exact convergence rate, the critical mini-batch size) for Theorem 4.8.

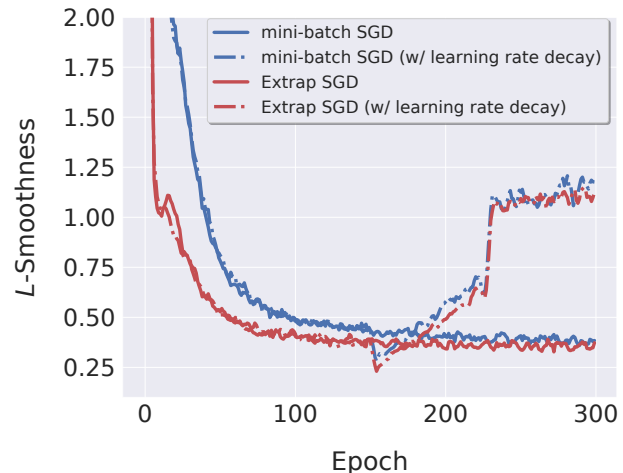
5. Experiments

We first briefly outline the general experimental setup below (for more details refer to Appendix C) and then thoroughly evaluate our framework on different challenging large-batch training tasks. We limit our attention to three standard and representative benchmarking tasks, with a controlled epoch budget (for each task). We ensure the used mini-batch size is a significant fraction of the whole dataset. Performing experiments on a much larger dataset for the same demonstration purposes is out of our computational ability³.

³ E.g., we use the mini-batch size of 8,192 (roughly 16% of the total data), out of 50,000 samples for CIFAR as our main tool of justification. While for ImageNet (Russakovsky et al., 2015) with 1.28 million data samples in total, the same fraction would result in roughly 800 workers for local mini-batch of size 256.



(a) Learning curves for different methods (w/o learning rate decay).



(b) Better smoothness of EXTRAP-SGD.

Figure 2. Understanding the learning behaviors of different methods on the large-batch training (with mini-batch size 8, 192 on 32 workers) for training ResNet-20 on CIFAR-10. The visualization of smoothness in Figure 2(b) follows the idea in (Santurkar et al., 2018; Haruki et al., 2019); it takes 8 additional steps (each with 30% of the update) in the direction of the update for each training step, and the smoothness of a training step is expressed by the maximum value of L (evaluated after local update steps) satisfying the Assumption 3. We use the learning rate scaling and warmup in Goyal et al. (2017) for the first 5 epochs, and γ and $\tilde{\gamma}$ are fine-tuned for different methods.

5.1. Experimental Setup

Datasets. We evaluate all methods on the following three tasks: (1) Image Classification for CIFAR-10/100 (Krizhevsky & Hinton, 2009) (50K training samples and 10K testing samples with 10/100 classes) with the standard data augmentation and preprocessing scheme (He et al., 2016; Huang et al., 2016b); (2) Language Modeling for WikiText2 (Merity et al., 2016) (the vocabulary size is 33K, and its train and validation set have 2 million tokens and 217K tokens respectively); and (3) Neural Machine Translation for Multi30k (Elliott et al., 2016).

Models and training schemes. Several benchmarking models are used in our experimental evaluation. (1) ResNet-20 (He et al., 2016) and VGG-11 (Simonyan & Zisserman, 2014) on CIFAR for image classification, (2) two-layer LSTM (Merity et al., 2017) with hidden dimension of size 128 on WikiText-2 for language modeling, and (3) a down-scaled transformer (factor of 2 w.r.t. the base model in Vaswani et al. (2017)) for neural machine translation. Weight initialization schemes for the three tasks follow Goyal et al. (2017); He et al. (2015), Merity et al. (2017) and Vaswani et al. (2017) respectively.

We use mini-batch SGD with a Nesterov momentum of 0.9 without dampening for image classification and language modeling tasks, and Adam for neural machine translation tasks. In the following experiment section, the term “mini-batch SGD” indicates the mini-batch SGD with Nesterov momentum unless mentioned otherwise.

For experiments on image classification and language modeling, unless mentioned otherwise the models are trained for 300 epochs; the local mini-batch sizes are set to 256 and 64 respectively. By default, all related experiments will use learning rate scaling and warmup scheme⁴ (Goyal et al., 2017; Hoffer et al., 2017). The learning rate is always gradually warmed up from a relatively small value for the first few epochs. Besides, the learning rate γ in image classification task will be dropped by a factor of 10 when the model has accessed 50% and 75% of the total number of training samples (He et al., 2016; Huang et al., 2016a). The LARS is only applied on image classification task⁵ (You et al., 2017).

For experiments on neural machine translation, we use standard inverse square root learning rate schedule (Vaswani et al., 2017). The warmup step is set to 4000 for the mini-batch size of 64 and will be linearly scaled down by the global mini-batch size⁶.

We carefully tune the learning rate γ and the trust term $\tilde{\gamma}$ in You et al. (2017). The tuning procedure ensures that the best hyper-parameter lies in the middle of our search grids; otherwise, we extend our search grid. The procedure of hyperparameter tuning can be found in Appendix C.1.

⁴ Since we will fine-tune the (to be scaled) learning rate, there is no difference between learning rate linear scaling (Goyal et al., 2017) and square root scaling (Hoffer et al., 2017) in our case.

⁵ Our implementation relies on the PyTorch extension of NVIDIA apex for mixed precision and distributed training.

⁶ We follow the practical instruction from NVIDIA.

5.2. Evaluation on Large-batch Training

Superior performance of EXTRAP-SGD on different tasks. We evaluate our extrapolation framework and compare it with SOTA large-batch training methods on CIFAR-10 image classification (Figure 2) and WikiText2 language modeling (Figure 3). To better exhibit the optimization behaviors of different methods, for these two tasks in this section we do not decay the learning rate. *The extrapolated SGD framework in general significantly accelerates the optimization and leads to better test performance than the existing SOTA methods.* For example, the smaller gradient Lipschitz constant illustrated in Figure 2(b) demonstrates the improved optimization landscape, which explains the at least $2\times$ speedup in terms of the convergence (after thorough hyperparameter tuning) in Figure 2(a) and Figure 3.

We further extend⁷ the extrapolation idea of EXTRAP-SGD to EXTRAP-ADAM and validate its effectiveness (compared with Adam) on neural machine translation with Transformer. The algorithmic description refers to Algorithm 3 in Appendix D.2. Figure 4 shows the results of large-batch training (using 4% and 16% of the training data per mini-batch) and EXTRAP-ADAM *again outperforms the Adam with at least $2\times$ speedup.*

Optimization v.s. Generalization benefits. To better understand when and how EXTRAP-SGD (and extrapolated SGD) help, we switch our attention to the commonly accepted training practices: using an initial large learning rate and decaying when the training plateaus. The common beliefs⁸ (LeCun et al., 1991; Kleinberg et al., 2018) argue that, the initial large learning rate accelerates the transition from random initialization to convergence regions (optimization), and the decaying leads the convergence to local minimum (generalization).

Table 1 thoroughly evaluates the large-batch training performance (ResNet-20 on CIFAR-10 for mini-batch size 8,192, with learning rate decay schedule) for all related methods. Though the remarkable optimization improvements in Figure 2(a) justify *the effects of EXTRAP-SGD (and extrapolated SGD), i.e. smooth the ill-conditioned loss landscape*, decaying the learning rate diminishes our advantages, as illustrated in Table 1 and Figure 6 in Appendix. *We argue that EXTRAP-SGD and its variants can smoothen the loss surface thus avoid some bad local minima regions, but it cannot guarantee to converge to a much better local minimum*⁹,

⁷ It is non-trivial to adapt Adam to the noise variants of the extrapolated SGD.

⁸ Recent work (Li et al., 2019; You et al., 2020a) complement the understanding of this phenomenon from the learning of different patterns via different learning rate scales; we leave the connection to this aspect for future work.

⁹ Additional experiments show that switching EXTRAP-SGD to mini-batch SGD after the first learning rate decay has similar

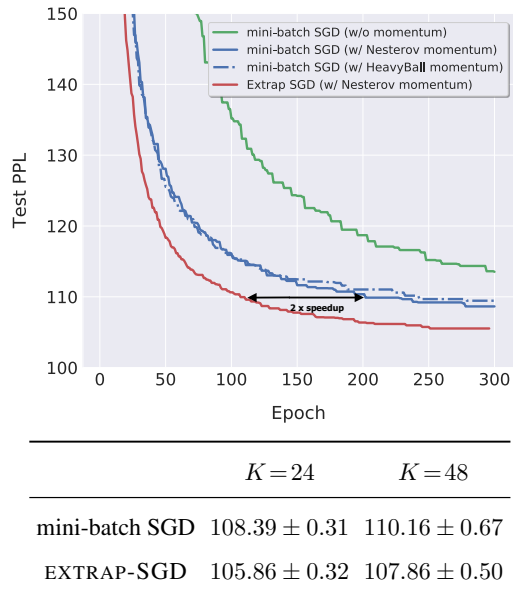


Figure 3. The perplexity (PPL, the lower the better) of training LSTM on WikiText-2. The global mini-batch size are 1,536 and 3,072 for $K = 24$ and $K = 48$ respectively, accounting for 2% and 4% of the total training data. We use the learning rate scaling and warmup in Goyal et al. (2017), and use constant learning rate after the warmup. We finetune the γ for mini-batch SGD (and its momentum variants); EXTRAP-SGD reuses the hyper-parameter from mini-batch SGD. The results of the inline table are averaged over three different seeds. The displayed learning curves are based on $K = 24$ and more details refer to Appendix E.2.

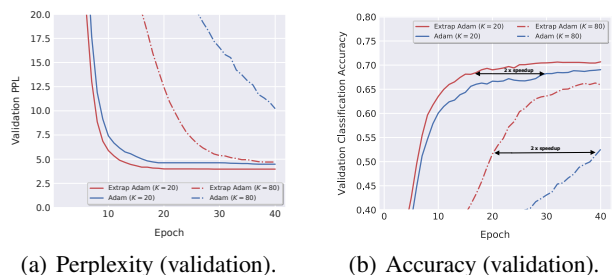


Figure 4. Training EXTRAP-ADAM on Multi30k with Transformer. The evaluations are performed on the validation dataset, for the training on $K = 20$ and $K = 80$ workers (corresponding to roughly 4% and 16% of the total training data). We use the standard inverse square root learning rate schedule (Vaswani et al., 2017) and scale the warmup step based on the number of workers.

contrary to the statements in (Wen et al., 2018; Haruki et al., 2019). They use (limited) empirical generalization metrics as the main arguments of the “flatter minima”, ignoring the complex training dynamics; their ignored SOTA optimization techniques (e.g. LARS in some experiments) might also result in the improper understandings. Our empirical results provide insights: *the primary benefits of EXTRAP-SGD are test performance as using EXTRAP-SGD alone.*

Table 1. The performance comparison of different methods for large-batch training on CIFAR-10 (mini-batch of size 8,192 on $K=32$). Two neural architectures (ResNet-20 and VGG-11) are considered (with and without batch normalization). Unless mentioned otherwise each method will use the learning rate scaling and warmup in Goyal et al. (2017) and LARS in You et al. (2017). We finetune γ and $\tilde{\gamma}$ for mini-batch SGD and SmoothOut (with $\tilde{\gamma}$ additionally). For the results of extrapolated SGD, we reuse the optimal γ and $\tilde{\gamma}$ tuned on mini-batch SGD; $\hat{\gamma}$ is fine-tuned for noise-based extrapolation variants. The results are averaged over three different seeds.

	mini-batch SGD (w/o LARS)	EXTRAP-SGD (w/o LARS)	mini-batch SGD	SmoothOut (Wen et al., 2018)	extrapolated SGD, uniform noise	extrapolated SGD, stochastic noise	EXTRAP-SGD
ResNet-20 on CIFAR-10	90.00 \pm 0.48	90.47 \pm 0.16	91.36 \pm 0.19	91.55 \pm 0.20	91.53 \pm 0.25	91.66 \pm 0.24	91.72 \pm 0.11
VGG-11 on CIFAR-10	73.09 \pm 9.35	76.79 \pm 3.5	86.64 \pm 0.10	86.92 \pm 0.15	87.00 \pm 0.31	86.04 \pm 0.43	87.00 \pm 0.26

on the optimization phase (early training phase) and will diminish in the later training phase. It is also reflected in Figure 2(b) in terms of the smoothness.

Combining EXTRAP-SGD with post-local SGD. Given our new insights in the previous paragraph, here we try to understand the importance of better optimization brought by EXTRAP-SGD for the eventual generalization. We consider the post-local SGD in Lin et al. (2020), a known technique arguing to converge to flatter local minimum for better generalization. This choice comes from the noticeable optimization benefits of EXTRAP-SGD in the initial training phase while post-local SGD targeting to converge to “flatter minima” for the later training phase. Please refer to Algorithm 2 in Appendix D.1 for training details.

We argue that EXTRAP-SGD biases the optimization trajectory towards a better-conditioned loss surface, where solutions with good generalization properties can be found more easily. Figure 5 challenges the extreme large-batch training (mini-batch of size 16,384 on 64 workers, accounting for 33% of training data) for ResNet-20 on CIFAR-10 with different epoch budgets. We can notice that the improper optimization in the critical initial learning phase of the mini-batch SGD results in a significant generalization gap, which cannot be addressed by adding post-local SGD or increasing the number of training epochs alone. EXTRAP-SGD, on the contrary, avoids the regions containing bad local minima, complementing the ability of post-local SGD for converging to better solutions. Table 2 additionally reports a similar observation for mini-batch of size 8,192 on more datasets.

5.3. Ablation study

EXTRAP-SGD for different local mini-batch sizes and number of workers. Table 3 in the Appendix E.1 evaluates how different combinations of the local mini-batch size and the number of workers will impact the performance, for a given global mini-batch size (8,192 for ResNet-20 on CIFAR-10). The benefits of EXTRAP-SGD can be further pronounced when increasing the number of workers, which is the common practice for large-batch training. Similar observation can be found in Figure 4 for the increased number of workers (as well as the global mini-batch size).

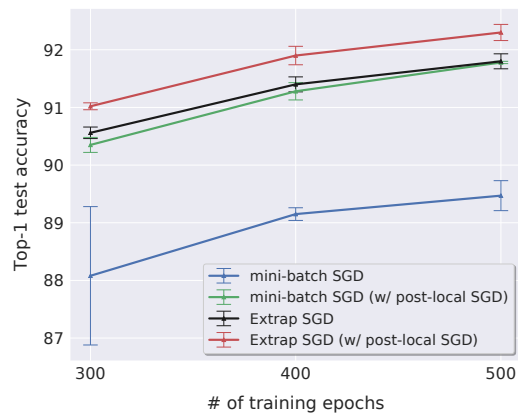


Figure 5. The test top-1 accuracy of integrating EXTRAP-SGD with post-local SGD (Lin et al., 2020). The performance of ResNet-20 on CIFAR-10 is evaluated with a global mini-batch size of 16,384 (33% of the total training data with 64 workers). By default we use the learning rate scaling and warmup in Goyal et al. (2017) and LARS in You et al. (2017). We individually finetune γ and $\tilde{\gamma}$ for each base method; the local update step H is tuned and set to $H=4$. The learning rate is decayed by 10 when the model has accessed 50% and 75% of the total training samples. The results are averaged over three different seeds.

Understanding the effect of momentum. Given the mixed effects of local extrapolation and momentum acceleration in EXTRAP-SGD, Figure 8 in Appendix E.3 decouples these two factors for the training ResNet-20 on CIFAR-10 with mini-batch of size 8,192. We can witness that (1) EXTRAP-SGD can take advantages of both extrapolation as well as the acceleration from the momentum, and thus can always be applied for accelerated and stabilized distributed optimization; (2) using extrapolation alone (no momentum) in EXTRAP-SGD still outperforms mini-batch SGD with tuned momentum factor; (3) tuning momentum factor for mini-batch SGD can marginally improve the optimization performance but cannot eliminate the optimization difficulty.

6. Conclusion

In this work, we adopt the idea of randomized smoothing to distributed training and propose EXTRAP-SGD to perform extrapolation with past local mini-batch gradients. The

Table 2. The test top-1 accuracy of integrating EXTRAP-SGD with post-local SGD (Lin et al., 2020). The performance of ResNet-20 on CIFAR-10/100 is evaluated with global mini-batch size 8, 192 ($K = 32$). By default we use the learning rate scaling and warmup in Goyal et al. (2017) and LARS in You et al. (2017). We individually finetune γ and $\tilde{\gamma}$ for each method; the local update step H tuned from $\{4, 8, 16\}$ (as in Lin et al. (2020)) in general improves the performance and we report the best performance with $H = 8$. The results are averaged over three different seeds.

	mini-batch SGD	mini-batch SGD (with post-local SGD)	EXTRAP-SGD	EXTRAP-SGD (with post-local SGD)
CIFAR-10	91.36 \pm 0.19	91.73 \pm 0.25	91.72 \pm 0.11	92.23 \pm 0.02
CIFAR-100	65.79 \pm 0.46	67.39 \pm 0.18	66.63 \pm 0.32	68.06 \pm 0.21

idea further extends to a unified framework, covering multiple noise extrapolation variants. We provide convergence guarantees for methods within this framework, and empirically justify the remarkable benefits of our methods on image classification, language modeling and neural machine translation tasks. We further investigate and understand the properties of our methods; the algorithms smoothen the ill-conditioned loss landscape for faster optimization and biases the optimization trajectory to well-conditioned regions. These insights further motivate us to combine our methods with post-local SGD for SOTA large-batch training.

Acknowledgements

We acknowledge funding from SNSF grant 200021_175796, as well as a Google Focused Research Award.

References

Achille, A., Rovere, M., and Soatto, S. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=BkeStsCckQ>.

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

Chavdarova, T., Gidel, G., Fleuret, F., and Lacoste-Julien, S. Reducing noise in gan training with variance reduced extragradient. *arXiv preprint arXiv:1904.08598*, 2019.

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

Diakonikolas, J. and Orecchia, L. Accelerated extra-gradient descent: A novel accelerated first-order method. *arXiv preprint arXiv:1706.04680*, 2017.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

Duchi, J. C., Bartlett, P. L., and Wainwright, M. J. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

Elliott, D., Frank, S., Sima’an, K., and Specia, L. Multi30k: Multilingual english-german image descriptions. *arXiv preprint arXiv:1605.00459*, 2016.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. *arXiv preprint arXiv:1802.10026*, 2018.

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

Gidel, G., Berard, H., Vignoud, G., Vincent, P., and Lacoste-Julien, S. A variational inequality perspective on generative adversarial networks. *arXiv preprint arXiv:1802.10551*, 2018.

Golatkar, A. S., Achille, A., and Soatto, S. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Advances in Neural Information Processing Systems*, pp. 10677–10687, 2019.

Golmant, N., Vemuri, N., Yao, Z., Feinberg, V., Gholami, A., Rothauge, K., Mahoney, M. W., and Gonzalez, J. On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941*, 2018.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Haruki, K., Suzuki, T., Hamakawa, Y., Toda, T., Sakai, R., Ozawa, M., and Kimura, M. Gradient noise convolution (gnc): Smoothing loss function for distributed large-batch sgd, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on

- imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.
- Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *arXiv preprint arXiv:1705.08741*, 2017.
- Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016a.
- Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pp. 646–661. Springer, 2016b.
- Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho, K., and Geras, K. The break-even point on the optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=r1g87C4KwB>.
- Juditsky, A., Nemirovski, A., and Tauvel, C. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.
- Kleinberg, R., Li, Y., and Yuan, Y. An alternative view: When does sgd escape local minima?, 2018.
- Korpelevich, G. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.
- Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.
- LeCun, Y., Kanter, I., and Solla, S. A. Second order properties of error surfaces: Learning time and generalization. In *Advances in neural information processing systems*, pp. 918–924, 1991.
- Leng, C., Li, H., Zhu, S., and Jin, R. Extremely low bit neural network: Squeeze the last bit out with admm, 2017.
- Li, H., Xu, Z., Taylor, G., and Goldstein, T. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.
- Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pp. 11669–11680, 2019.
- Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don’t use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=BleyO1BFPr>.
- Ma, S., Bassily, R., and Belkin, M. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. *arXiv preprint arXiv:1712.06559*, 2017.
- Masters, D. and Luschi, C. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.
- McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.
- Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.
- Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *arXiv preprint*, pp. arXiv:1708.02182, 2017.
- Mishchenko, K., Kovalev, D., Shulgin, E., Richtárik, P., and Malitsky, Y. Revisiting stochastic extragradient. *arXiv preprint arXiv:1905.11373*, 2019.
- Nemirovski, A. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1): 229–251, 2004.
- Nesterov, Y. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pp. 372–376, 1983.
- Nesterov, Y. Random gradient-free minimization of convex functions. CORE Discussion Papers 2011001, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2011. URL <https://EconPapers.repec.org/RePEc:cor:louvco:2011001>.

- Neyshabur, B. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.
- Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pp. 2483–2493, 2018.
- Scaman, K., Bach, F., Bubeck, S., Massoulié, L., and Lee, Y. T. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pp. 2740–2749, 2018.
- Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training, 2018.
- Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Stich, S. U. and Karimireddy, S. P. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350*, 2019.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- Wang, W. and Srebro, N. Stochastic nonconvex optimization with large minibatches. *arXiv preprint arXiv:1709.08728*, 2017.
- Wen, W., Wang, Y., Yan, F., Xu, C., Wu, C., Chen, Y., and Li, H. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv preprint arXiv:1805.07898*, 2018.
- Xu, Y., Yuan, Z., Yang, S., Jin, R., and Yang, T. On the convergence of (stochastic) gradient descent with extrapolation for non-convex minimization. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Aug 2019. doi: 10.24963/ijcai.2019/556. URL <http://dx.doi.org/10.24963/ijcai.2019/556>.
- Yan, Y., Yang, T., Li, Z., Lin, Q., and Yang, Y. A unified analysis of stochastic momentum methods for deep learning. In *IJCAI*, pp. 2955–2961, 2018. URL <https://doi.org/10.24963/ijcai.2018/410>.
- Yao, Z., Gholami, A., Lei, Q., Keutzer, K., and Mahoney, M. W. Hessian-based analysis of large batch training and robustness to adversaries. *arXiv preprint arXiv:1802.08241*, 2018.
- Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. Gradient diversity: a key ingredient for scalable distributed learning. *arXiv preprint arXiv:1706.05699*, 2017.
- You, K., Long, M., Wang, J., and Jordan, M. I. How does learning rate decay help modern neural networks? 2020a. URL <https://openreview.net/forum?id=r1eOnh4YPB>.
- You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.
- You, Y., Hseu, J., Ying, C., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large-batch training for lstm and beyond. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, 2019.
- You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020b. URL <https://openreview.net/forum?id=Syx4wnEtvH>.
- Yu, H., Jin, R., and Yang, S. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817*, 2019.
- Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S. J., Kumar, S., and Sra, S. Why adam beats sgd for attention models. *arXiv preprint arXiv:1912.03194*, 2019.