Large Batch Optimization for Deep Learning Using New Complete Layer-Wise Adaptive Rate Scaling

Zhouyuan Huo[1], **Bin Gu**[2, 3], **Heng Huang**[3, 4*]
[1] Google, USA;
[2] MBZUAI, United Arab Emirates
[3] JD Finance America Corporation, Mountain View, CA, USA
[4] Electrical and Computer Engineering, University of Pittsburgh, PA, USA
zhouyuan.huo@gmail.com, jsgubin@gmail.com, heng.huang@pitt.edu

Abstract

Training deep neural networks using a large batch size has shown promising results and benefits many real-world applications. Warmup is one of nontrivial techniques to stabilize the convergence of large batch training. However, warmup is an empirical method and it is still unknown whether there is a better algorithm with theoretical underpinnings. In this paper, we propose a novel Complete Layer-wise Adaptive Rate Scaling (CLARS) algorithm for large-batch training. We prove the convergence of our algorithm by introducing a new fine-grained analysis of gradient-based methods. Furthermore, the new analysis also helps to understand two other empirical tricks, layer-wise adaptive rate scaling and linear learning rate scaling. We conduct extensive experiments and demonstrate that the proposed algorithm outperforms gradual warmup technique by a large margin and defeats the convergence of the state-of-the-art large-batch optimizer in training advanced deep neural networks (ResNet, DenseNet, MobileNet) on ImageNet dataset.

Introduction

Deep learning has made significant breakthroughs in many fields, such as computer vision (He et al. 2016, 2017; Krizhevsky, Sutskever, and Hinton 2012; Ren et al. 2015), nature language processing (Devlin et al. 2018; Hochreiter and Schmidhuber 1997; Vaswani et al. 2017), and reinforcement learning (Mnih et al. 2013; Silver et al. 2017). Recent studies show that better performance can usually be achieved by training a larger neural network with a bigger dataset (Mahajan et al. 2018; Radford et al. 2019). Nonetheless, it is time-consuming to train deep neural networks, which limits the efficiency of deep learning research. For example, training ResNet50 on ImageNet with batch size 256 needs to take about 29 hours to obtain 75.3% Top-1 accuracy on 8 Tesla P100 GPUs (He et al. 2016). Thus, it is a critical topic to reduce the training time for the development of deep learning using data parallelism (Dean et al. 2012; Krizhevsky 2014; Yadan et al. 2013) or model parallelism (Huang et al. 2019; Huo et al. 2018). However, the large-batch neural network training using conventional gradient-based methods techniques usually requires heuristic tricks and leads to worse generalization errors (Hoffer, Hubara, and Soudry 2017; Keskar et al. 2016).

Many empirical training techniques have been proposed for large-batch deep learning optimization. (Goyal et al. 2017) proposed to adjust the learning rate through linear learning rate scaling and gradual warmup. By using these two techniques, they successfully trained ResNet50 with a batch size of 8192 on 256 GPUs in one hour with no loss of accuracy. Finding that the ratios of weight's ℓ_2-norm to gradient's ℓ_2-norm vary greatly among layers, (You, Gitman, and Ginsburg 2017; You et al. 2019a) proposed and analyzed the state-of-the-art large-batch optimizer Layer-wise Adaptive Rate Scaling (LARS) and scaled the batch size to 16384 for training ResNet50 on ImageNet. However, LARS still requires warmup in early epochs of training and may diverge if it is not tuned properly. There are many theoretical analysis about linear learning rate scaling (Lian et al. 2015, 2016; Zhang et al. 2019). However, it is still unknown whether there is a better algorithm than warmup trick for large batch training with theoretical underpinnings.

In this paper, we target to remove the empirical warmup trick for large-batch training and propose a better algorithm with theoretical underpins. We summarize our main contributions as follows:

1. We propose a novel Complete Layer-wise Adaptive Rate Scaling (CLARS) algorithm for large-batch deep neural networks optimization, which provides an superior performance than warmup trick in the beginning of training.
2. We analyze the convergence of the proposed CLARS algorithm by introducing a new fine-grained analysis for gradient-based methods and demonstrate that warmup and CLARS alleviate the training difficulties caused by layer-wise gradient variance. Furthermore, our analysis can help to understand layer-wise adaptive rate scaling and linear learning rate scaling.
3. Extensive experimental results demonstrate that the proposed CLARS method outperforms gradual warmup by a large margin and defeats the convergence of the state-of-the-art large-batch optimizer in training advanced deep neural networks (ResNet, DenseNet, MobileNet) on ImageNet dataset.

*Corresponding Author

Preliminaries

Gradient-Based Methods: The loss function of a neural network is minimizing the average loss over a dataset of n samples:

$$\min_{w\in\mathbb{R}^d} \quad \{f(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w)\}, \tag{1}$$

where d denotes the dimension of the neural network. Momentum-based methods have been widely used in deep learning optimization, especially computer vision, and obtain state-of-the-art results (He et al. 2016; Huang et al. 2016). According to (Nesterov 1983), mini-batch Nesterov Accelerated Gradient (mNAG) optimizes the problem (1) as follows:

$$\begin{aligned} v_{t+1} &= w_t - \gamma\frac{1}{B}\sum_{i\in I_t}\nabla f_i(w_t), \\ w_{t+1} &= v_{t+1} + \beta(v_{t+1} - v_t), \end{aligned} \tag{2}$$

where I_t is the mini-batch samples with $|I_t| = B$, γ is the learning rate, $\beta \in [0, 1)$ is the momentum constant and v is the momentum vector. When $\beta = 0$, Eq. (2) represents the procedures of mini-batch Gradient Descent (mGD). Learning rate γ is scaled up linearly when batch size B is large (Goyal et al. 2017). However, using a learning rate γ for all layers may lead to performance degradation (You, Gitman, and Ginsburg 2017).

Layer-Wise Learning Rate Scaling: To train neural networks with large batch size, (You, Gitman, and Ginsburg 2017; You et al. 2019b) proposed and analyzed Layer-Wise Adaptive Rate Scaling (LARS). Suppose a neural network has K layers, we can rewrite $w = [(w)_1, (w)_2, ..., (w)_K]$ with $(w)_k \in \mathbb{R}^{d_k}$ and $d = \sum_{k=1}^{K} d_k$. The learning rate at layer k is updated as follows:

$$\gamma_k \quad = \quad \gamma_{scale} \times \eta \times \frac{\|(w_t)_k\|_2}{\left\|\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)\right\|_2}, \tag{3}$$

where $\gamma_{scale} = \gamma_{base} \times \frac{B}{B_{base}}$ and $\eta = 0.001$ in (You, Gitman, and Ginsburg 2017). γ_{base} and B_{base} depends on model and dataset. For example, we set $\gamma_{base} = 0.1$ and $B_{base} = 128$ to train ResNet on CIFAR10. However, LARS should work together with warmup trick for large batch training. Otherwise, it even diverges in the beginning of training if warmup trick (Goyal et al. 2017) is absent.

Complete Layer-Wise Adaptive Rate Scaling

In this section, we propose to replace warmup trick with a novel Complete Layer-wise Adaptive Rate Scaling (CLARS) algorithm for large-batch deep learning optimization.

Define $U \in \mathbb{R}^{d\times d}$ as a permutation matrix where every row and column contains precisely a single 1 with 0s everywhere else. Let $U = [U_1, U_2, ..., U_K]$ and U_k corresponds to the parameters of layer k, the relation between w and w_k is $w = \sum_{k=1}^{K} U_k w_k$. Let $\nabla_k f_i(w_t)$ denote the stochastic gradient with respect to the parameters at layer k and γ_k denote its corresponding learning rate at layer k. Thus, Eq. (2) of mNAG with batch I_t can be rewritten as:

$$\begin{cases} v_{t+1} &= w_t - \sum_{k=1}^{K}\gamma_k U_k\left(\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)\right) \\ w_{t+1} &= v_{t+1} + \beta(v_{t+1} - v_t) \end{cases}. \tag{4}$$

At each iteration, the learning rate γ_k at layer k is updated using Complete Layer-wise Adaptive Rate Scaling (CLARS) as follows:

$$\gamma_k \quad = \quad \gamma_{scale} \times \eta \times \frac{\|(w_t)_k\|_2}{\frac{1}{B}\sum_{i\in I_t}\|\nabla_k f_i(w_t)\|_2}, \tag{5}$$

where $\gamma_{scale} = \gamma_{base} \times \frac{B}{B_{base}}$ and $\eta \in \{10^{-3}, 10^{-2}, 10^{-1}\}$ is a constant. γ_{base} and B_{base} are user prescribed parameters according to the model and dataset. For example, we set $\gamma_{base} = 0.1$ and $B_{base} = 128$ to train ResNet on CIFAR10. To obtain a clear understanding of Eq. (5), we can rewrite it as follows:

$$\begin{aligned} \gamma_k \quad = \quad & \gamma_{scale} \times \eta \times \frac{\|(w_t)_k\|_2}{\left\|\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)\right\|_2} \\ & \times \frac{\left\|\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)\right\|_2}{\frac{1}{B}\sum_{i\in I_t}\|\nabla_k f_i(w_t)\|_2}. \end{aligned}$$

It is easy to find out that Eq. (5) is equal to multiplying the LARS learning rate in Eq. (3) with a new term $\frac{\left\|\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)\right\|_2}{\frac{1}{B}\sum_{i\in I_t}\|\nabla_k f_i(w_t)\|_2}$, which plays a critical role in removing the warmup. The proposed CLARS algorithm for gradient-based methods is briefly summarized in Algorithm 1.

In the following section, we will show that warmup trick and CLARS are both targeted to alleviate the training difficulties caused by **Gradient Variance**. CLARS algorithm approximates the gradient variance properly through $\frac{\left\|\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)\right\|_2}{\frac{1}{B}\sum_{i\in I_t}\|\nabla_k f_i(w_t)\|_2}$, and is well supported theoretically. In the experimental section, we will also visualize the variation of gradient variance at different layers for different neural networks and demonstrate that it is an important factor to be considered to accelerate the convergence.

Fine-Grained Convergence Analysis

Fine-Grained Micro-Steps and Assumptions

In this section, we propose a new fine-grained method for the convergence analysis of gradient-based methods. Based on the fine-grained analysis, we prove the convergence rate of mini-batch Gradient Descent (mGD) and mini-batch Nesterov's Accelerated Gradient (mNAG) for deep learning problems. More insights are obtained by analyzing their convergence properties.

Each step of mNAG in Eq. (4) can be regarded as the result of updating v, w for K micro-steps, where the gradient at each micro-step is $\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)$. At micro-step $t{:}s$, we have layer index $k(s) = s \pmod K + 1$. For example, when $s = 0$, we are updating the parameters of layer

Algorithm 1 Complete Layer-Wise Adaptive Rate Scaling

Require: γ_{scale}: Maximum learning rate
Require: β: Momentum parameter
Require: $\eta = 0.01$
for $t = 0, 1, 2, \cdots, T-1$ **do**
  Sample large-batch I_t randomly with batch size B;
  Compute large-batch gradient $\frac{1}{B}\sum_{i\in I_t}\nabla f_i(w_t)$;
  Compute the average of gradient norm for K layers $\frac{1}{B}\sum_{i\in I_t}\|\nabla_k \nabla f_i(w_t)\|_2^2$;
  Update layer-wise learning rate γ_k following Eq. (5);
  Update the model w_t and momentum term v_t following Eq. (4);
end for
Output w_T as the final result.

$k(0) = 1$. Defining $w_{t:0} = w_t$, $w_{t:K} = w_{t+1}$, we can obtain Eq. (4) after applying following equations from $s = 0$ to $s = K - 1$:

$$\begin{cases} v_{t:s+1} &= w_{t:s} - \frac{\gamma_k}{B}\sum\limits_{i\in I_t} U_k \nabla_k f_i(w_t) \\ w_{t:s+1} &= v_{t:s+1} + \beta(v_{t:s+1} - v_{t:s}) \end{cases}. \quad (6)$$

Following the idea of block-wise Lipschitz continuous assumption in (Beck and Tetruashvili 2013) and regarding layers as blocks, we suppose that two layer-wise assumptions are satisfied for any K-layer neural network throughout this paper, .

Assumption 1 (Layer-Wise Lipschitz Continuous Gradient) *Assume that the gradient of f is layer-wise Lipschitz continuous and the Lipschitz constant corresponding to layer k is L_k for any layer $k \in \{1, 2, ..., K\}$. For any $w \in \mathbb{R}^d$ and $v = [v_1, v_2, ..., v_K] \in \mathbb{R}^d$, the following inequality is satisfied that for any $k \in \{1, 2, ..., K\}$:*

$$\|\nabla_k f(w) - \nabla_k f(w + U_k v_k)\|_2 \quad \leq \quad L_k \|v_k\|_2.$$

In addition, we also assume that there is a "global" Lipschitz constant L_g such that:

$$\|\nabla f(w) - \nabla f(w + v)\|_2 \leq L_g \|v\|_2.$$

Lipschitz constants L_k of different layers are not equal and can be affected by multiple factors, for example, position (top or bottom) or layer type (CNN or FCN). (Zou, Balan, and Singh 2018) estimated Lipschitz constants empirically and verified that Lipschitz constants of gradients at different layers vary a lot. L_k represents the property at layer k and plays an essential role in tuning learning rates.

Assumption 2 (Layer-Wise Bounded Variance) *Assume that the variance of stochastic gradient with respect to the parameters of layer k is upper bounded. For any $k \in \{1, 2, ..., K\}$ and $w \in \mathbb{R}^d$, there exists $M_k > 0$ and $M > 0$ so that:*

$$\mathbb{E}\left\|\nabla_k f_i(w) - \nabla_k f(w)\right\|_2^2 \quad \leq \quad M_k \mathbb{E}\|\nabla_k f(w)\|_2^2 + M.$$

M_k presents the variation of gradient variance at layer k of the neural network. In the analysis, we will show that the upper bound of learning rate at layer k is dependent on the value of M_k. In the experimental section, we will also show that M_k varies greatly in different layers and taking it into consideration can greatly accelerate the convergence of neural networks training.

Difficulties of Convergence Analysis: There are two major difficulties in proving the convergence rate using the proposed fine-grained micro-steps. (I) Micro-step induces stale gradient in the analysis. At each micro-step $t{:}s$ in Eq. (6), gradient is computed using the stale model w_t, rather than the latest model $w_{t:s}$. (II) K Lipschitz constants and bounded variance for K layers are considered separately and simultaneously, which are much more complicated than just considering a single L_g or $M_g = \max M_k$ for the whole model.

Convergence Guarantees of Two Gradient-Based Methods

Based on the proposed fine-grained analysis, we prove that both of mini-batch Gradient Descent (mGD) and mini-batch Nesterov's Accelerated Gradient (mNAG) admit sub-linear convergence guarantee $O\left(\frac{1}{\sqrt{T}}\right)$ for non-convex problems. Finally, we obtain some new insights about the gradient-based methods by taking mNAG as an example. At first, we let $\beta = 0$ in Eq. (4) and Eq. (6), and analyze the convergence of mGD method.

Theorem 1 (Convergence of mGD) *Under Assumptions 1 and 2, let $f_{\inf}$ denote the minimum value of problem $f(w)$, $\kappa_k = \frac{L_g}{L_k} \leq \kappa$, $\gamma_k = \frac{\gamma}{L_k}$, and $\sum_{k=1}^K q_k \mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2$ represents the expectation of $\mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2$ with probability $q_k = \frac{1/L_k}{\sum_{k=1}^K (1/L_k)}$ for any k. As long as $\gamma_k \leq \min\left\{\frac{1}{8L_k}, \frac{B}{8L_k M_k}\right\}$ and $\frac{1}{K}\sum\limits_{k=1}^K \gamma_k \leq \min\left\{\frac{1}{2L_g}, \frac{1}{2L_g}\sqrt{\frac{B}{M_g}}\right\}$, it is guaranteed that:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K} q_k \mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2 \leq \frac{8(f(w_0) - f_{\inf})}{T\gamma \sum\limits_{k=1}^{K}\frac{1}{L_k}} + C_B.$$

where $C_B = \frac{(4+2\kappa)M\gamma}{B}$.

In Theorem 1, we use $\sum\limits_{k=1}^K q_k \mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2$ to measure convergence in the paper. Specially, if $L_k = L_g$ for all k, it is easy to know that $q_k = \frac{1}{K}$ for all k and $\sum\limits_{k=1}^K q_k \mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2 = \frac{1}{K}\mathbb{E}\left\|\nabla f(w_t)\right\|_2^2$ which is similar to the criterion in (Yang, Lin, and Li 2016). So far, we have proved the convergence of mGD method for non-convex problems. When $\beta \neq 0$, we can also prove the convergence of mNAG as follows:

From Theorem 1, we prove that mGD admits sub-linear convergence rate $O\left(\frac{1}{\sqrt{T}}\right)$ for non-convex problems.

Corollary 1 (Sub-Linear Convergence Rate of mGD) *Theorem 1 is satisfied and follow its notations. Suppose* $\frac{1}{8L_k}$ *dominates the upper bound of* γ_k, $\frac{1}{K}\sum_{k=1}^{K}\gamma_k \leq \min\left\{\frac{1}{2L_g}, \frac{1}{2L_g}\sqrt{\frac{B}{M_g}}\right\}$, *and* $\gamma = \min\left\{\frac{1}{8}, \sqrt{\frac{B(f(w_0)-f_{\inf})}{TM\sum_{k=1}^{K}\frac{1}{L_k}}}\right\}$, *mGD is guaranteed to converge that:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K} q_k \mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2 \leq \frac{64(f(w_0)-f_{\inf})}{T\sum_{k=1}^{K}\frac{1}{L_k}} + (12+2\kappa)\sqrt{\frac{M(f(w_0)-f_{\inf})}{TB\sum_{k=1}^{K}\frac{1}{L_k}}}.$$

Then, we also analyze the convergence rate of Nesterov's Accelerated Gradient method, which shows better convergence empirically optimizing deep neural networks.

Theorem 2 (Convergence of mNAG) *Under Assumptions 1 and 2, let* $f_{\inf}$ *denote the minimum value of problem* $f(w)$, $\kappa_k = \frac{L_g}{L_k} \leq \kappa$, $\gamma_k = \frac{\gamma}{L_k}$, *and* $\sum_{k=1}^{K} q_k \mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2$ *represents the expectation of* $\mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2$ *with probability* $q_k = \frac{1/L_k}{\sum_{k=1}^{K}(1/L_K)}$ *for any* k. *Therefore, as long as* $\gamma_k \leq \min\left\{\frac{(1-\beta)}{8L_k}, \frac{(1-\beta)B}{8L_k M_k}\right\}$ *and* $\frac{1}{K}\sum_{k=1}^{K}\gamma_k \leq \min\left\{\frac{(1-\beta)^2}{4\beta^2 L_g}, \frac{(1-\beta)^2\sqrt{B}}{4\beta^2 L_g\sqrt{M_g}}, \frac{(1-\beta)\sqrt{B}}{4L_g\sqrt{M_g}}, \frac{(1-\beta)}{4L_g}\right\}$, *it is satisfied that:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K} q_k \mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2 \leq \frac{8(1-\beta)(f(w_0)-f_{\inf})}{T\gamma\sum_{k=1}^{K}\frac{1}{L_k}} + \frac{M\gamma}{(1-\beta)B}\left(4+2\kappa+\frac{2\kappa}{(1-\beta)}\right).$$

From Theorem 2, we can easily prove that mNAG is guaranteed to converge for non-convex problems with a sub-linear rate $O\left(\frac{1}{\sqrt{T}}\right)$ as follows:

Corollary 2 (Sub-Linear Convergence of mNAG) *Theorem 2 is satisfied and follow its notations, Suppose* $\frac{1-\beta}{8L_k}$ *dominates the upper bound of* γ_k, $\frac{1}{K}\sum_{k=1}^{K}\gamma_k \leq \min\left\{\frac{(1-\beta)^2}{4\beta^2 L_g}, \frac{(1-\beta)^2\sqrt{B}}{4\beta^2 L_g\sqrt{M_g}}, \frac{(1-\beta)\sqrt{B}}{4L_g\sqrt{M_g}}, \frac{(1-\beta)}{4L_g}\right\}$, *and* $\gamma = \min\left\{\frac{1-\beta}{8}, \sqrt{\frac{B(f(w_0)-f_{\inf})}{TM\sum_{k=1}^{K}\frac{1}{L_k}}}\right\}$, *mNAG is guaranteed to converge that:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K} q_k \mathbb{E}\left\|\nabla_k f(w_t)\right\|_2^2 \leq \frac{64(f(w_0)-f_{\inf})}{(1-\beta)T\sum_{k=1}^{K}\frac{1}{L_k}} + \left(8 + \frac{1}{(1-\beta)}\left(4+2\kappa+\frac{2\kappa}{(1-\beta)}\right)\right)\sqrt{\frac{M(f(w_0)-f_{\inf})}{TB\sum_{k=1}^{K}\frac{1}{L_k}}}. \quad (7)$$

Similarly, we know that the result in (Yang, Lin, and Li 2016) is a special case of Theorem 2 when $L_k = L_g$ and $M_k = M_g$.

Corollary 3 (Convergence when $L_k = L_g$ and $M_k = M_g$) *Suppose Theorem 2 is satisfied and follow its notations. If* $L_k = L_g$, *and* $M_k = M_g$, $M_C = KM$, *we have* $\kappa_k = 1$, $\gamma_g = \gamma_k$. *As long as the learning rate* $\gamma_g \leq \min\left\{\frac{1-\beta}{8L_g}, \frac{B(1-\beta)}{8L_g M_g}, \frac{(1-\beta)\sqrt{B}}{4L_g\sqrt{M_g}}, \frac{(1-\beta)^2\sqrt{B}}{4\beta^2 L_g\sqrt{M_g}}, \frac{(1-\beta)}{4\beta^2 L_g}\right\}$, *it is guaranteed that:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(w_t)\right\|_2^2 \leq \frac{8(1-\eta)(f(w_0)-f_{\inf})}{T\gamma_g} + \frac{M_C L_g \gamma_g}{(1-\beta)}\left(6+\frac{2}{1-\beta}\right) \quad (8)$$

Discussions About the Convergence of mNAG

According our fine-grained convergence analysis of gradient-based methods, we take mNAG as an example and gain more insights about the convergence of mNAG for neural networks.

Layer-Wise Gradient Variance Factor M_k. Define M_k as the gradient variance factor at layer k, which is dependent on the data and the model, and varies in the process of training. Because of the upper bound of $\gamma_k \leq \min\left\{\frac{(1-\beta)}{8L_k}, \frac{(1-\beta)B}{8L_k M_k}\right\}$ in Theorem 2, it shows that batch size B can be scaled up as long as $B \leq M_k$. Therefore, a larger M_k helps the algorithm obtain faster speedup. Besides, if B is fixed, it also denotes that a large M_k leads to a smaller learning rate γ_k with convergence guarantee. In the following section, we will show that warmup is closely related to M_k.

Layer-Wise Scaled Learning Rate. From Theorem 2, we know that the upper bound of learning rate γ_k at each layer is dependent on $\frac{1}{L_k}$. LARS (You, Gitman, and Ginsburg 2017) scales the learning rate of each layer adaptively at step t by multiplying $\frac{\|(w_t)_k\|_2}{\|\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)\|_2}$ in Eq. (3). From Assumption 1, we can think of LARS as scaling the learning rate at layer k by multiplying the approximation of $\frac{1}{L_k} \approx \frac{\|(w_t)_k\|_2}{\|\nabla_k f(w_t)\|_2}$, where we make $v_k = 0$ and $w_t + U_k v_k = 0$. Therefore, the procedure of LARS is consistent with our theoretical analysis in Theorem 2 that learning rate of layer k is dependent on the Lipschitz constant at this layer $\gamma_k = \frac{\gamma}{L_k}$.

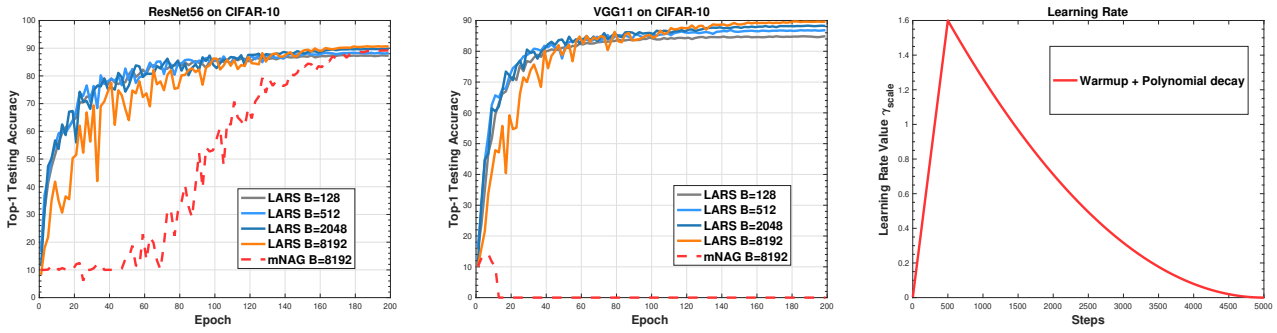

Figure 1: Training loss and Top-1 testing accuracy of training ResNet56 and VGG11 (with batch normalization layer) on CIFAR-10. Batch size B scales up from $B = 128$ to $B = 8192$. The right figure presents the variation of γ_{scale} in Eq. (3) when $B = 2048$.

We compare LARS with mNAG using a large batch size. Results in Figure 1 demonstrate that LARS converges much faster than mNAG when $B = 8192$. mNAG even diverges in training VGG11 using CIFAR-10. In the experiments, $\gamma_{base} = 0.1$, $B_{base} = 128$, and $\eta = 0.001$ for LARS algorithm.

Linear Learning Rate Scaling. Data parallelism is widely used in the training of deep learning models, and linear speedup can be obtained if learning rate and communication can be properly handled. According to Theorem 2, if the upper bound of learning rate γ_k is dominated by $\frac{(1-\beta)B}{8L_k M_k}$, it is easy to know that increasing mini-batch size B will also increase the upper bound linearly. Therefore, it is reasonable to use scale up the learning rate linearly in large batch training.

Experimental Results

In this section, we conduct large-batch deep learning training experiments to validate our analysis empirically and demonstrate the superior performance of CLARS method over warmup trick. All experiments are implemented in PyTorch 1.0 (Paszke et al. 2017) with Cuda v10.0 and performed on a machine with Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz and 4 Tesla P40 GPUs.

Variations of Gradient Variance

The gradual warmup was essential for large-batch deep learning optimization because linearly scaled γ_{scale} can be so large that the loss cannot converge in early epochs (Goyal et al. 2017). In the gradual warmup, γ_{scale} is replaced with a small value at the beginning and increased back gradually after a few epochs.

According to our analysis in Section , we guess that the gradual warmup is to compensate the effect of $\frac{1}{M_k}$ in the upper bound of learning rate. We train 5-layer FCN, 5-layer CNN on MNIST (LeCun et al. 1998) and ResNet8 on CIFAR-10 using mNAG for 6-10 epochs. Constant learning rate 0.001 is used for all layers and batch size $B = 128$. After each epoch, we approximate the gradient variance factor M_k by computing the ratio of $\frac{1}{n}\sum_{i=1}^{n}\|\nabla_k f_i(w_t)\|_2^2$ to $\|\frac{1}{n}\sum_{i=1}^{n}\nabla_k f_i(w_t)\|_2^2$ on training data. Figure 2 presents the variation of M_k at each layers. It is obvious that M_k of top layers are larger than the values of M_k at lower layers, which indicates that the upper bound of learning rate at top layers are smaller. Thus, smaller learning rates should be used on top layers at early epochs. Our observation matches the result in (Gotmare et al. 2018) that freezing fully connected layers at early epochs allows for comparable performance with warmup.

Comparison with Related Methods

We evaluate the proposed Algorithm 1 by conducting extensive experiments. Current Pytorch library does not support computing the individual gradient's norm in parallel in the process of gradient computation. To reduce the time consumption in computing M_k, we approximate it using $M_k \approx \frac{\|\frac{1}{B}\sum_{i\in I_t}\nabla_k f_i(w_t)\|_2^2}{\frac{1}{|J_t|}\sum_{j\in J_t}\|\nabla_k f_j(w_t)\|_2^2}$, where $|J_t| = 512$ for all experiments in this section. The numerator is known after the gradient computation, and the denominator is obtained in a small size. Since $|J_t| \ll B$, the computational time of approximating M_k can be ignored when the computation is amortized on multiple devices.

In Figure 3, we make a comparison of warmup trick, fully-connected layer freezing (which freezes the update of fully-connected layers in first few epochs) (Gotmare et al. 2018), and CLARS (layer-wise warmup). We train ResNet56 and VGG11 (with batch normalization layer) on CIFAR-100 using mNAG ($\beta = 0.9$) with batch size $B = 8192$ for 200 epochs. Learning rate starts from 1.0 and multiplies by 0.1 at epochs 60, 120 and 150. The compared methods only take effect in the first 20 epochs. Standard data preprocessing techniques are used as in (He et al. 2016). Visualization in Figure 3 shows that CLARS always outperforms other compared methods. FC freezing works well in the beginning for VGG11, while it diverges after that.

We also evaluate CLARS algorithm by training ResNet50, DenseNet121, and MobileNetv2 on ImageNet (Deng et al. 2009). Because there are not enough GPUs to compute 16384 gradients at one time, we set batch size $B = 512$ and accumulate the gradients for 32 steps

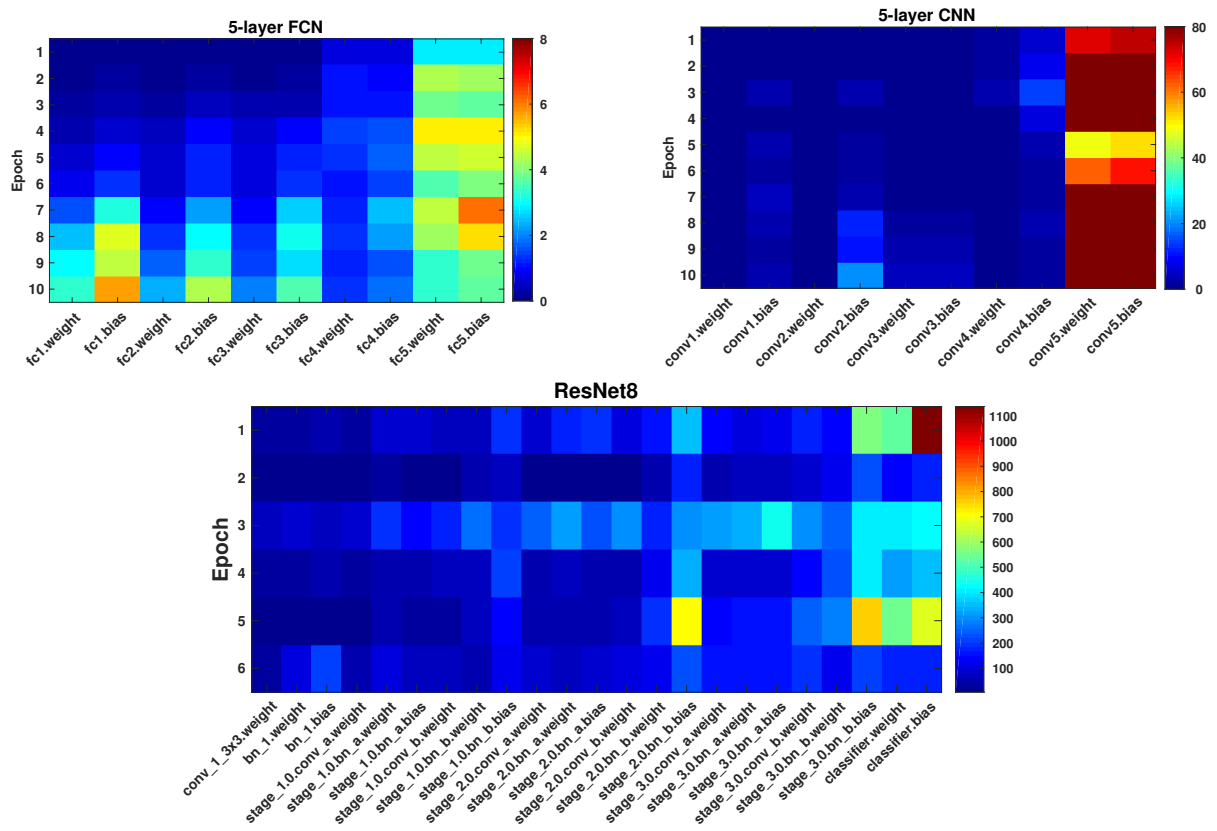

Figure 2: Visualizations of the variation of M_k. We train 5-layer FCN, 5-layer CNN with sigmoid activation on MNIST for 10 epochs, and ResNet8 on CIFAR-10 for 6 epochs. Empirical results show that top layers have higher value of M_k than bottom layers in the first few epochs.

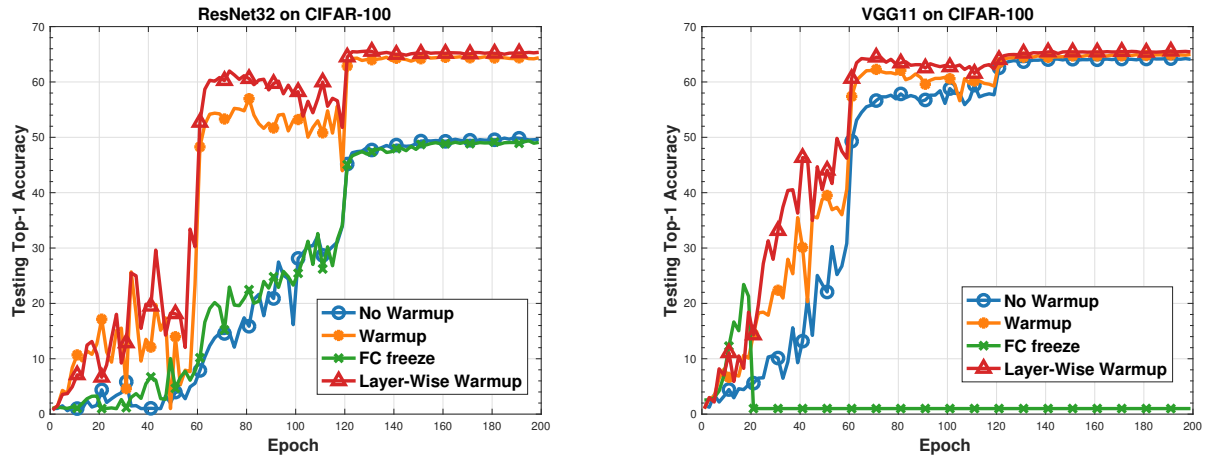

Figure 3: Testing Top-1 accuracy of warmup trick, fully-connected layer freezing (FC freeze), and CLARS (layer-wise warmup). We optimize ResNet56 and VGG11 on CIFAR-100 with batch size $B = 4096$ for 200 epochs. We apply warmup trick and FC freeze in the first 20 epochs and keep the learning rate constants afterwards.

before updating the model as (You, Gitman, and Ginsburg 2017). Following the official implementation in (You, Gitman, and Ginsburg 2017), we set $\eta = 10^{-3}$ for LARS, $\gamma_{scale} = 25.0$ for $B = 16384$ and adjust the learning rate using 5-epoch warmup and polynomial decay. For CLARS, there is no warmup and we set $\eta = 10^{-2}$ (LARS always

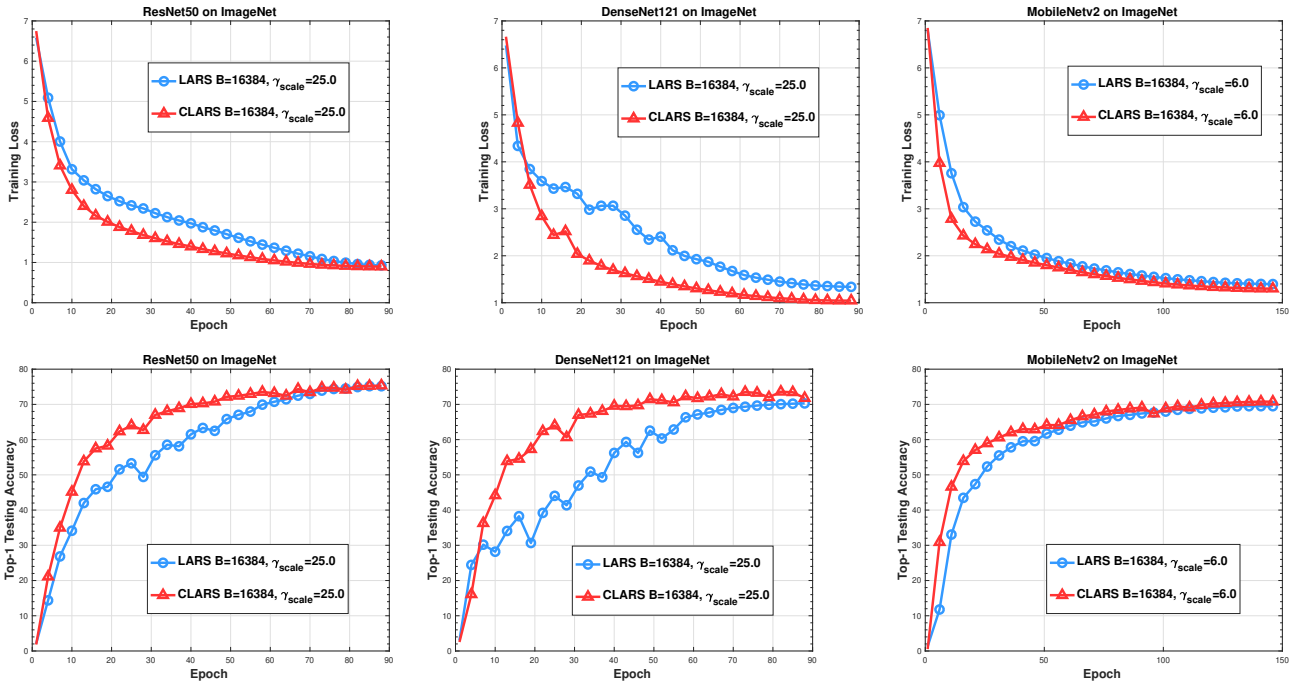

Figure 4: Comparison between LARS and CLARS methods on training ResNet50, DenseNet121, and MobileNetv2 for ImageNet. We train ResNet50, DenseNet121 for 90 epochs with batch size $B = 16384$ and $\gamma_{scale} = 25.0$. MobileNetv2 is trained for 150 epochs with batch size $B = 16384$ and $\gamma_{scale} = 6.0$.

Model	small batch	LARS	CLARS
ResNet50	75.3	75.1	75.1
DenseNet121	75.0	70.4	**73.6**
MobileNetv2	72.0	69.6	**70.9**

Table 1: Comparison of Top-1 Testing Accuracy between small batch training, and two large batch training using LARS and CLARS on ImageNet with batch size $B = 16384$ for 90 epochs.

diverges with this value). Experimental results in Figure 4 present that CLARS algorithm always converges much faster than the state-of-the-art large-batch optimizer LARS with warmup trick on advanced neural networks. Besides, CLARS can obtain better test error than LARS as in Table 1, which demonstrates that it has a better generalization performance than the compared method.

Conclusion

In this paper, we proposed a novel Complete Layer-wise Adaptive Rate Scaling (CLARS) algorithm to remove warmup in the large-batch deep learning training. After that, we introduced fine-grained analysis and prove the convergence of the proposed algorithm for non-convex problems. We proved its convergence by introducing a new fine-grained analysis of gradient-based methods. Furthermore, the new analysis also helps to understand two other empirical tricks, layer-wise adaptive rate scaling and linear learning rate scaling. Extensive experiments demonstrate that the proposed algorithm outperforms warmup trick by a large margin and defeats the convergence of the state-of-the-art large-batch optimizer (LARS) in training advanced deep neural networks on ImageNet dataset.

Acknowledgements

This work was partially supported by NSF IIS 1845666, 1852606, 1838627, 1837956, 1956002, 2040588.

References

Beck, A.; and Tetruashvili, L. 2013. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization* 23(4): 2037–2060.

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Senior, A.; Tucker, P.; Yang, K.; Le, Q. V.; et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*, 1223–1231.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 248–255. Ieee.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Gotmare, A.; Keskar, N. S.; Xiong, C.; and Socher, R. 2018. A Closer Look at Deep Learning Heuristics: Learn-

ing rate restarts, Warmup and Distillation. *arXiv preprint arXiv:1810.13243* .

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

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

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

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780.

Hoffer, E.; Hubara, I.; and Soudry, D. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, 1731–1741.

Huang, G.; Liu, Z.; Weinberger, K. Q.; and van der Maaten, L. 2016. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993* .

Huang, Y.; Cheng, Y.; Bapna, A.; Firat, O.; Chen, D.; Chen, M.; Lee, H.; Ngiam, J.; Le, Q. V.; Wu, Y.; et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, 103–112.

Huo, Z.; Gu, B.; Yang, Q.; and Huang, H. 2018. Decoupled parallel backpropagation with convergence guarantee. *arXiv preprint arXiv:1804.10574* .

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836* .

Krizhevsky, A. 2014. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997* .

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

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

Lian, X.; Huang, Y.; Li, Y.; and Liu, J. 2015. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, 2737–2745.

Lian, X.; Zhang, H.; Hsieh, C.-J.; Huang, Y.; and Liu, J. 2016. A comprehensive linear speedup analysis for asynchronous stochastic parallel optimization from zeroth-order to first-order. In *Advances in Neural Information Processing Systems*, 3054–3062.

Mahajan, D.; Girshick, R.; Ramanathan, V.; He, K.; Paluri, M.; Li, Y.; Bharambe, A.; and van der Maaten, L. 2018. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 181–196.

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

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

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1: 8.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676): 354.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Yadan, O.; Adams, K.; Taigman, Y.; and Ranzato, M. 2013. Multi-gpu training of convnets. *arXiv preprint arXiv:1312.5853* .

Yang, T.; Lin, Q.; and Li, Z. 2016. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv preprint arXiv:1604.03257* .

You, Y.; Gitman, I.; and Ginsburg, B. 2017. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888* 6.

You, Y.; Hseu, J.; Ying, C.; Demmel, J.; Keutzer, K.; and Hsieh, C.-J. 2019a. Large-Batch Training for LSTM and Beyond. *arXiv preprint arXiv:1901.08256* .

You, Y.; Li, J.; Hseu, J.; Song, X.; Demmel, J.; and Hsieh, C.-J. 2019b. Reducing BERT Pre-Training Time from 3 Days to 76 Minutes. *arXiv preprint arXiv:1904.00962* .

Zhang, G.; Li, L.; Nado, Z.; Martens, J.; Sachdeva, S.; Dahl, G.; Shallue, C.; and Grosse, R. B. 2019. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, 8194–8205.

Zou, D.; Balan, R.; and Singh, M. 2018. On Lipschitz Bounds of General Convolutional Neural Networks. *arXiv preprint arXiv:1808.01415* .