Distributed Adversarial Training to Robustify Deep Neural Networks at Scale

Gaoyuan Zhang[1,*] **Songtao Lu**[1,*] **Yihua Zhang**[2] **Xiangyi Chen**[3] **Pin-Yu Chen**[1] **Quanfu Fan**[1] **Lee Martie**[1]
Lior Horesh[1] **Mingyi Hong**[3] **Sijia Liu**[1,2]

[1]IBM Research
[2]Michigan State University
[3]University of Minnesota
[*]Equal Contribution

Abstract

Current deep neural networks (DNNs) are vulnerable to adversarial attacks, where adversarial perturbations to the inputs can change or manipulate classification. To defend against such attacks, an effective and popular approach, known as *adversarial training (AT)*, has been shown to mitigate the negative impact of adversarial attacks by virtue of a min-max robust training method. While effective, it remains unclear whether it can successfully be adapted to the distributed learning context. The power of distributed optimization over multiple machines enables us to scale up robust training over large models and datasets. Spurred by that, we propose *distributed adversarial training (DAT)*, a *large-batch* adversarial training framework implemented over multiple machines. We show that DAT is general, which supports training over labeled and unlabeled data, multiple types of attack generation methods, and gradient compression operations favored for distributed optimization. Theoretically, we provide, under standard conditions in the optimization theory, the convergence rate of DAT to the first-order stationary points in general non-convex settings. Empirically, we demonstrate that DAT either matches or outperforms state-of-the-art robust accuracies and achieves a graceful training speedup (e.g., on ResNet–50 under ImageNet). Codes are available at `https://github.com/dat-2022/dat`.

1 INTRODUCTION

The rapid increase of research in DNNs and their adoption in practice is, in part, owed to the significant breakthroughs made with DNNs in computer vision [Alom et al., 2018]. Yet, with the apparent power of DNNs, there remains a serious weakness of robustness. That is, DNNs can easily be manipulated (by an adversary) to output drastically different classifications and can be done so in a controlled and directed way. This process is known as an adversarial attack and considered as one of the major hurdles in using DNNs in security critical and real-world applications [Goodfellow et al., 2015, Szegedy et al., 2013, Carlini and Wagner, 2017, Papernot et al., 2016, Kurakin et al., 2016, Eykholt et al., 2018, Xu et al., 2019b].

Methods to train DNNs being robust against adversarial attacks are now a major focus in research [Xu et al., 2019a]. But most of them are far from satisfactory [Athalye et al., 2018] with the exception of the adversarial training (AT) approach [Madry et al., 2017]. AT is a min-max robust training method that minimizes the worst-case training loss at adversarially perturbed examples. AT has inspired a wide range of state-of-the-art defenses [Zhang et al., 2019b, Sinha et al., 2018, Boopathy et al., 2020, Carmon et al., 2019, Shafahi et al., 2019, Zhang et al., 2019a], which ultimately resort to min-max optimization. However, different from standard training, AT is more computationally intensive and is difficult to scale.

Motivation and challenges. *First*, although a 'fast' version of AT (we call Fast AT) was developed in [Wong et al., 2020] where an iterative inner maximization solver is replaced by a simplified (single-step) solution, it may suffer several problems compared to AT: unstable robust learning performance [Li et al., 2020], over-sensitive to learning rate schedule [Rice et al., 2020], and catastrophic forgetting of robustness against strong attacks [Andriushchenko and Flammarion, 2020]. As a result, AT is still the dominant robust training protocol across applications. Spurred by that, we propose DAT, a new approach to speed up AT by allowing for scaling batch size with distributed machines. *Second*, existing AT-type methods are generally built on *centralized* optimization. The need of AT in a *distributed* setting arises when centralized robust training becomes infeasible or ineffective. For example, training data are distributed as they

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

cannot centrally be stored at a single machine due to their size or privacy. Or computing units are distributed as they allow large-batch optimization to improve the scalability of training.

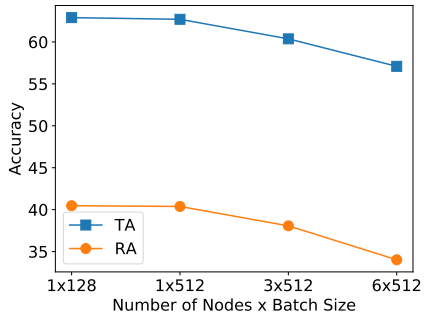


Figure 1: Robust accuracy (RA) and standard test accuracy (TA) of AT vs. scaled batch size under (ImageNet, ResNet-50) using distributed machines.

While designing a distributed solution is important, doing so effectively is non-trivial. Figure 1 demonstrates an example: When scaling batch size with the number of computing nodes, the conventional AT method yields a large performance drop in both robust and standard accuracies. Thus, the adaptation of AT to distributed learning leaves many unanswered questions. In this work, we aim to design a principled and theoretically-grounded (large-batch) DAT framework by making full use of the computing capability of multiple data-locality (distributed) machines, and show that DAT expands the capacity of data storage and the computational scalability. Furthermore, due to the existence of many variants of AT, it requires a careful and systematic study on distributed AT in its formulation, methodology, theory and performance evaluation.

Contributions. We list our main contributions below.

(i) We provide a general algorithmic framework for DAT, which supports multiple (large-batch) distributed variants of AT, e.g., supervised AT and semi-supervised AT.

(ii) In theory, we quantify how descent errors from multiple sources (gradient estimation, quantization, adaptive learning rate, and inner maximization oracle) affect the convergence of DAT. We prove that the convergence speed of DAT to the first-order stationary points in general non-convex settings at a rate of $O(1/\sqrt{T})$, where T is the total number of iterations. This result matches the standard convergence rate of classic training algorithms, e.g., stochastic gradient descent (SGD), for only the minimization problems.

(iii) In practice, we make a comprehensive empirical study on DAT, showing its effectiveness to (1) robust training over ImageNet, (2) provably robust training by randomized smoothing, (3) robust training with unlabeled data, (4) robust pretraining + finetuning, and (5) robust training across different computing and communication configurations.

2 RELATED WORK

Training robust classifiers. AT [Madry et al., 2017], the first known min-max optimization-based defense, has inspired a wide range of other effective defenses. Examples include adversarial logit pairing [Kannan et al., 2018], input gradient or curvature regularization [Ross and Doshi-Velez, 2018, Moosavi-Dezfooli et al., 2019], trade-off between robustness and accuracy (TRADES) [Zhang et al., 2019b], distributionally robust training [Sinha et al., 2018], dynamic adversarial training [Wang et al., 2019b], robust input attribution regularization [Boopathy et al., 2020], certifiably robust training [Wong and Kolter, 2017], and semi-supervised robust training [Stanforth et al., 2019, Carmon et al., 2019].

In particular, some recent works proposed *fast but approximate* AT algorithms, such as 'free' AT [Shafahi et al., 2019], you only propagate once (YOPO) [Zhang et al., 2019a], and fast gradient sign method (FGSM) based AT [Wong et al., 2020]. These algorithms achieve speedup in training by simplifying the inner maximization step of AT, but are designed for centralized model training. A few works made empirical efforts to scale AT up by using multiple computing nodes [Xie et al., 2019, Kang et al., 2019, Qin et al., 2019], they were limited to specific use cases and lacked a thorough study on when and how distributed learning helps, either in theory or in practice.

Distributed model training. Distributed optimization has been found to be effective for the standard training of machine learning models [Dean et al., 2012, Goyal et al., 2017, You et al., 2019, Chen et al., 2020]. In contrast to centralized optimization, distributed learning enables increasing the batch size proportional to the number of computing nodes/machines. However, it is challenging to train a model via large-batch optimization without incurring accuracy loss compared to the standard training with same number of epochs [Krizhevsky, 2014, Keskar et al., 2016]. To tackle this challenge, it was shown in [You et al., 2017b, 2018, 2019] that adaptation of learning rates to the increase of the batch size is an essential mean to boost the performance of large-batch optimization. A layer-wise adaptive learning rate strategy was then proposed to speed up the training as well as preserve the accuracy. Although these works have witnessed several successful applications of distributed learning in training *standard* image classifiers, they leave the question of how to build *robust* DNNs with DAT open. In this paper, we show that the power of layer-wise adaptive learning rate also applies to DAT. Since distributed learning introduces machine-machine communication overhead, another line of work [Alistarh et al., 2017, Yu et al., 2019, Bernstein et al., 2018, Wangni et al., 2018, Stich et al., 2018, Wang et al., 2019a] focused on the design of communication-

efficient distributed optimization algorithms.

The study on distributed learning is extensive, but the problem of distributed min-max optimization is less explored, with some exceptions [Srivastava et al., 2011, Notarnicola et al., 2018, Hanada et al., 2017, Tsaknakis et al., 2020, Liu et al., 2019a,b]. A key difference to our work is that none of the aforementioned literature studied the *large-batch min-max optimization* with its applications to training *robust* DNNs, neither theoretically nor empirically. While there are recent proposed algorithms for training Generative Adversarial Nets (GANs) [Liu et al., 2019a,b], training robust DNNs against adversarial examples is intrinsically different from GAN training. In particular, training robust DNNs requires inner maximization with respect to each training data rather than empirical maximization with respect to model parameters. Such an essential difference leads to different optimization goals, algorithms, convergence analyses and implementations.

3 PROBLEM FORMULATION

In this section, we first review the standard setup of adversarial training (AT) [Madry et al., 2017], and then propose a general min-max setup for distributed AT (DAT).

Adversarial training. AT [Madry et al., 2017] is a min-max optimization method for training robust ML/DL models against adversarial examples [Goodfellow et al., 2015]. Formally, AT solves the problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}\left[\underset{\|\boldsymbol{\delta}\|_\infty\leq\epsilon}{\text{maximize}}\ \ell(\boldsymbol{\theta},\mathbf{x}+\boldsymbol{\delta};y)\right], \quad (1)$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ denotes the vector of model parameters, $\boldsymbol{\delta} \in \mathbb{R}^n$ is the vector of input perturbations within an ℓ_∞ ball of the given radius ϵ, namely, $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$, $(\mathbf{x}, y) \in \mathcal{D}$ corresponds to the training example $\mathbf{x}$ with label y in the dataset $\mathcal{D}$, and ℓ represents a pre-defined training loss, e.g., the cross-entropy (CE) loss. The rationale behind problem (1) is that the model $\boldsymbol{\theta}$ is robustly trained against the *worst-case* loss induced by the adversarially perturbed samples. It is worth noting that the AT problem (1) is *different* from conventional stochastic min-max optimization problems, e.g., GANs training [Goodfellow et al., 2014]. Note that in (1), the stochastic sampling corresponding to the expectation over $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ is conducted *prior to* the inner maximization operation. Such a difference leads to the *sample-specific* adversarial perturbation $\boldsymbol{\delta}(\mathbf{x}) := \text{maximize}_{\|\boldsymbol{\delta}\|_\infty\leq\epsilon}\ \ell(\boldsymbol{\theta},\mathbf{x}+\boldsymbol{\delta};y)$.

Distributed AT (DAT). Let us consider a popular parameter-server model of distributed learning [Dean et al., 2012]. Formally, there exist M workers each of which has access to a local dataset $\mathcal{D}^{(i)}$, and thus $\mathcal{D} = \cup_{i=1}^{M}\mathcal{D}^{(i)}$. There also exists a server/master node (e.g., one of workers could perform as server), which collects local information (e.g., individual gradients) from the other workers to update the model parameters $\boldsymbol{\theta}$. Spurred by (1), DAT solves problems of the following generic form,

$$\begin{aligned}&\underset{\boldsymbol{\theta}}{\text{minimize}}\ \frac{1}{M}\sum_{i=1}^{M} f_i(\boldsymbol{\theta};\mathcal{D}^{(i)}),\\ &f_i =: \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{(i)}}\left[\lambda\ell(\boldsymbol{\theta};\mathbf{x},y)+\max_{\|\boldsymbol{\delta}\|_\infty\leq\epsilon}\phi(\boldsymbol{\theta},\boldsymbol{\delta};\mathbf{x},y)\right]\end{aligned} \quad (2)$$

where f_i denotes the local cost function at the ith worker, ϕ is a robustness regularizer against the input perturbation $\boldsymbol{\delta}$, and $\lambda \geq 0$ is a regularization parameter that strikes a balance between the training loss and the worst-case robustness regularization. In (2), if $M = 1$, $\mathcal{D}^{(1)} = \mathcal{D}$, $\lambda = 0$ and $\phi = \ell$, then the DAT problem reduces to the AT problem (1). We cover two categories of (2). ① DAT with labeled data: In (2), we consider $\phi(\boldsymbol{\theta}, \boldsymbol{\delta}; \mathbf{x}, y) = \ell(\boldsymbol{\theta}, \mathbf{x} + \boldsymbol{\delta}; y)$ with labeled training data $(\mathbf{x}, y) \in \mathcal{D}^{(i)}$ for $i \in [M]$. Here $[M]$ denotes the integer set $\{1, 2, \ldots, M\}$. ② DAT with unlabeled data: In (2), different from DAT with labeled data, we augment $\mathcal{D}^{(i)}$ with an unlabeled dataset, and define the robust regularizer ϕ as the pseudo-labeled worst-case CE loss [Carmon et al., 2019] or the TRADES regularizer [Stanforth et al., 2019, Zhang et al., 2019b].

4 METHODOLOGIES

At the first glance, distributed learning seems being naturally applied since problem (2) is decomposable over multiple workers. Yet, the actual case is much more complex. **First**, in contrast to standard AT, DAT allows for using a M times larger batch size to update the model parameters $\boldsymbol{\theta}$ in (2). Thus, given the same number of epochs, DAT takes M fewer gradient updates than AT. Although there exist some large-batch model training techniques for solving *min-only* problems [You et al., 2017a,b, 2018, 2019, Goyal et al., 2017, Keskar et al., 2016], it remains unclear if they are effective to DAT due to its *min-max* optimization nature. **Second**, either AT or distributed learning has its own challenges. In AT, for ease of attack generation, i.e., conducting inner maximization of (2), fast gradient sign method (FGSM) was leveraged to improve its computation efficiency [Wong et al., 2020]. In distributed learning, gradient compression [Alistarh et al., 2017, Yu et al., 2019] was used for reducing communication overhead. Thus, it also remains unclear whether these customizations are adaptable to DAT. In a nutshell, the distributed min-max optimization-based robust training algorithm has not been well studied previously, particularly in the use of different types of attack generators (inner maximization oracles), gradient quantization, large-batch size, and adaptive learning rate. Although either of the standalone techniques was studied separately,

justifying their coherent integration 'actually works' (both practically and theoretically) is quite demanding.

Algorithmic framework of DAT. DAT follows the framework of distributed learning with parameter server. In what follows, we elaborate on its key components through its meta-form shown by Algorithm 1 (see its detailed version in Algorithm A1). DAT contains three algorithmic blocks. In the *first* block, every distributed worker calls for a maximization oracle to obtain the adversarial perturbation for each sample within a data batch, then computes the gradient of the local cost function f_i in (2) with respect to (w.r.t.) model parameters $\boldsymbol{\theta}$. And every worker is allowed to quantize/compress the local gradient prior to transmission to the server. In the *second* block, the server aggregates the local gradients, and transmits the aggregated gradient (or the quantized gradient) to the other workers. In the *third* block, the model parameters are eventually updated by a minimization oracle at each worker based on the received gradient information from the server.

Algorithm 1 Meta-version of DAT (Alg. A1 in Supplement)

1: **for** Worker $i = 1, 2, \ldots, M$ **do** ▷ Block 1
2: Sample-wise attack generation (A1)
3: Local gradient computation (A2)
4: Worker-server communication
5: **end for**
6: Gradient aggregation at server (A3) ▷ Block 2
7: Server-worker communication
8: **for** Worker $i = 1, 2, \ldots, M$ **do** ▷ Block 3
9: Model parameter update (A4)
10: **end for**

Large-batch challenge in DAT and a layerwise adaptive learning rate (LALR) solution. In DAT, the aggregated gradient (Step 6 in Algorithm 1) is built on the data batch that is M times larger than the standard AT. This leads to a large-batch challenge in min-max optimization. This challenge can also be verified from Fig. 1. To overcome the large-batch challenge, we adopt the technique of layerwise adaptive learning rate (LALR), backed up by the recent successful applications to the standard training of large-scale image classification and language modeling networks with large data batch [You et al., 2019, 2017b].

To be more specific, the model training recipe using LALR becomes

$$\boldsymbol{\theta}_{t+1,i} = \boldsymbol{\theta}_{t,i} - \frac{\tau(\|\boldsymbol{\theta}_{t,i}\|_2) \cdot \eta_t}{\|\mathbf{u}_{t,i}\|_2} \cdot \mathbf{u}_{t,i}, \quad \forall i \in [h], \qquad (3)$$

where $\boldsymbol{\theta}_{t,i}$ denotes the ith-layer parameters at iteration t, with $\boldsymbol{\theta}_t = [\boldsymbol{\theta}_{t,1}^\top, \ldots, \boldsymbol{\theta}_{t,h}^\top]^\top$, h is the number of layers, $\mathbf{u}_t$ is a descent direction computed based on the first-order gradient w.r.t. model parameters $\boldsymbol{\theta}_t$, $\tau(\|\boldsymbol{\theta}_{t,i}\|_2) = \min\{\max\{\|\boldsymbol{\theta}_{t,i}\|_2, c_l\}, c_u\}$ is a *layerwise* scaling factor of the *adaptive* learning rate $\frac{\eta_t}{\|\mathbf{u}_{t,i}\|_2}$, and $c_l = 0$ and $c_u = 10$ are set in our experiments (see Appendix 4.1 for some ablation studies on hyperparameter selection).

In (3), the specific form of the descent direction $\mathbf{u}_t$ is determined by the optimizer employed. For example, if the adaptive momentum (Adam) method is used, then $\mathbf{u}_t$ is given by the exponential moving average of past gradients scaled by square root of exponential moving averages of squared past gradients [Reddi et al., 2018, Chen et al., 2018]. Such a variant of (3) that uses Adam as the base algorithm is also known as LAMB [You et al., 2019] in standard training. However, it was elusive if the advantage of LALR is preserved in large-batch min-max optimization. As will be evident later, the effectiveness of LALR in DAT can be justified from both theoretical and empirical perspectives. The rationale is that the layer-wise adaptive learning rate smooths the optimization trajectory so that a larger learning rate can be used without causing sharp optima even in distributed min-max optimization.

Other add-ons for DAT. In what follows, we illustrate two add-ons to improve computation and communication efficiency of DAT.

➢ *Inner maximization: Iterative vs. one-shot solution.* In DAT, each worker calls for an inner maximization oracle to generate adversarial perturbations (Step 2 of Algorithm 1). We specify two solvers of perturbation generation: iterative projected gradient descent (PGD) and one-shot (projected) FGSM [Goodfellow et al., 2015, Wong et al., 2020]. Our experiments will show that FGSM together with LALR works well in DAT. We also remark that other techniques [Shafahi et al., 2019, Zhang et al., 2019a] can also be used to simplify inner maximization, however, we focus on FGSM since it is computationally lightest.

➢ *Gradient quantization.* In contrast to standard AT, DAT may call for worker-server communications (Steps 4 and 7 of Algorithm 1). That is, if a single-precision floating-point data type is used, then DAT needs to transmit $32d$ bits per worker-server communication at each iteration. Recall that d is the dimension of $\boldsymbol{\theta}$. In order to reduce the communication cost, DAT has the option to quantize the transmitted gradients using a fixed number of bits fewer than 32. We specify the gradient quantization operation as the *randomized quantizer* [Alistarh et al., 2017, Yu et al., 2019]. In Sec. 6 we will show that DAT, combined with gradient quantization, still leads to a competitive performance. For example, the robust accuracy of ResNet-50 trained by a 8-bit DAT (performing quantization at Step 4 of Algorithm 1) for ImageNet is just 0.55% lower than the robust accuracy achieved by the 32-bit DAT. It is also worth mentioning that the All-reduce communication protocol can be regarded as a special case of the parameter-server setting in Algorithm 1 when every worker performs as a server. In this case, the

communication network becomes fully connected and only the worker-server communication (Step 4 of Algorithm 1) is needed. Please refer to Appendix 1 for more details on gradient quantization.

5 CONVERGENCE ANALYSIS OF DAT

Although standard AT has been proved with convergence guarantees [Wang et al., 2019b, Gao et al., 2019], none of existing work addressed the convergence of DAT and took into account LALR and gradient quantization, even in the standard AT setup. Different from AT, DAT needs to quantify the descent errors from multiple sources (such as gradient estimation, quantization, adaptive learning rate, and inner maximization oracle). Before showing the challenges of proving the convergence rate guarantees, we first give the following assumptions.

Assumptions. Defining $\Psi(\boldsymbol{\theta}) := \frac{1}{M}\sum_{i=1}^{M} f_i(\boldsymbol{\theta}; \mathcal{D}^{(i)})$ in (2), we measure the convergence of DAT by the first-order stationarity of Ψ. Prior to convergence analysis, we impose the following assumptions: ($\mathcal{A}1$) $\Psi(\boldsymbol{\theta})$ is with layer-wise Lipschitz continuous gradients; ($\mathcal{A}2$) ϕ) in (2) is strongly concave with respect to $\boldsymbol{\delta}$ and with Lipschitz continuous gradients within the perturbation constraint; ($\mathcal{A}3$) Stochastic gradient is unbiased and has bounded variance for each worker denoted by σ^2. Note that the validity of ($\mathcal{A}2$) could be justified from [Sinha et al., 2018, Wang et al., 2019b] by imposing a strongly convex regularization into the neighborhood of $\boldsymbol{\delta}$. $\mathcal{A}2$ is needed for tractability of analysis. We refer readers to Appendix 2.1 for more justifications on our assumptions ($\mathcal{A}1$)-($\mathcal{A}3$).

Technical challenges. In theory, the incorporation of LALR makes the analysis of min-max optimization highly non-trivial. The fundamental challenge lies in the nonlinear coupling between the biased adaptive gradient estimate resulted from LALR and the additional error generated from alternating update in DAT. From (3), we can see that the updated $\boldsymbol{\theta}$ is based on the normalized gradient, while if we perform convergence by applying the gradient Lipschitz continuity, the descent of the objective is measured by $\nabla\Psi(\boldsymbol{\theta}_t)$. This mismatch in the magnitude results in the bias term. The situation here is even worse, since the maximization problem cannot be solved exactly, the size of the bias depends on how close between the output of the oracle and the optimal solution w.r.t. $\boldsymbol{\delta}$ given $\boldsymbol{\theta}$.

We have proposed a new descent lemma (Lemma 2 in Appendix) to measure the decrease of the objective value in the context of alternative optimization, and showed that the bias error resulted from the layer-wise normalization can be compensated by large-batch training (Theorem 1). Prior to our work, we are *not* aware of any established convergence analysis for large-batch min-max optimization.

Convergence rate. In Theorem 1, we present the sublinear rate of DAT.

Theorem 1. *Suppose that assumptions $\mathcal{A}1$-$\mathcal{A}3$ hold, the inner maximizer of DAT provides a ε-approximate solution (i.e., the ℓ_2-norm of inner gradient is upper bounded by ε), and the learning rate is set by $\eta_t \sim \mathcal{O}(1/\sqrt{T})$, then $\{\boldsymbol{\theta}_t\}_{t=1}^{T}$ generated by DAT yields the convergence rate*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla_{\boldsymbol{\theta}}\Psi(\boldsymbol{\theta}_t)\|_2^2 = \mathcal{O}\left(\frac{1}{\sqrt{T}} + \frac{\sigma}{\sqrt{MB}} + \min\left\{\frac{d}{4^b}, \frac{\sqrt{d}}{2^b}\right\} + \varepsilon\right), \quad (4)$$

where b denotes the number of quantization bits, and $B = \min\{|\mathcal{B}_t^{(i)}|, \forall t, i\}$ stands for the smallest batch size per worker.

Proof: Please see Appendix 3. □

The error rate given by (4) involves four terms. The term $\mathcal{O}(1/\sqrt{MB})$ characterizes the benefit of using the large per-worker batch size B and M computing nodes in DAT. It is introduced since the variance of adaptive gradients (i.e., σ^2) is reduced by a factor $1/MB$, where $1/M$ corresponds to the linear speedup by M machines. In (4), the term $\min\{\frac{d}{4^b}, \frac{\sqrt{d}}{2^b}\}$ arises due to the variance of compressed gradients, and the other two terms imply the dependence on the number of iterations T as well as the ε-accuracy of the inner maximization oracle. We highlight that our convergence analysis (Theorem 1) is not barely a combination of LALR-enabled standard training analysis [You et al., 2019, 2017b] and adversarial training convergence analysis [Wang et al., 2019b, Gao et al., 2019]. Different from the previous work, we address the fundamental challenges in (a) quantifying the descent property of the objective value at the presence of multi-source errors during alternating min-max optimization, and (b) deriving the theoretical relationship between large data batch (across distributed machines) and the eventual convergence error of DAT.

6 EXPERIMENTS

We empirically evaluate DAT and show its success in training robust DNNs across multiple applications, which include ① adversarially robust ImageNet training, ② provably robust training by randomized smoothing, ③ semi-supervised robust training with unlabeled data, ④ robust transfer learning, ⑤ DAT using different communication protocols.

6.1 EXPERIMENT SETUP

DNN models and datasets. We use Pre-act ResNet-18 [He et al., 2016b] and ResNet-50 [He et al., 2016a] for image classification, where the former is shortened as ResNet-18. And we use ImageNet [Deng et al., 2009] for supervised DAT and augmented CIFAR-10 [Carmon et al., 2019] for semi-supervised DAT. In the latter setup, CIFAR-10 is augmented with unlabeled data drawn from 80 Million Tiny Images. When studying pre-trained model's transferability, CIFAR-100 is used as a target dataset for down-stream classification.

Computing resources. We train a DNN using p computing nodes, each of which contains q GPUs (Nvidia V100 or P100). Nodes are connected with 1Gbps ethernet. *A configuration of computing resources is noted by* $p \times q$. If $p > 1$, then the training is conducted in a *distributed* manner. And we split training data into p subsets, each of which is stored at a local node. We note that the batch size $6 \times 512 = 3072$ is used for ImageNet over 36 GPUs. Unless specified otherwise, DAT is conducted using Ring-AllReduce, which requires one-sided quantization in Step 4 of Algorithm 1.

Baseline methods. We consider 2 *variants* of DAT: 1) *DAT-PGD*, namely, DAT using (iterative) PGD as the inner maximization oracle; and 2) *DAT-FGSM*, namely, DAT using one-step (projected) FGSM [Wong et al., 2020] as the inner maximization oracle. Additionally, we consider 4 training *baselines*: 1) *AT* [Madry et al., 2017]; 2) *Fast AT* [Wong et al., 2020]; 3) *DAT w/o LALR*, namely, a distributed implementation of AT or Fast AT but *without* considering LALR; and 4) *DAT-LSGD* [Xie et al., 2019], namely, a distributed implementation of large-batch SGD (LSGD) for AT. We remark that conventional AT and Fast AT are centralized training methods.

Table 1: DAT (in gray color) on (ImageNet, ResNet-50), compared with baselines, in TA (%), RA (%), AA (%), communication time per epoch (seconds), and total training time (including communication time) per epoch (seconds). For brevity, '$p \times q$' represents '# nodes $\times$ # GPUs per node', 'C' represents communication time in seconds, and 'T' represents training time in seconds.

Method	ImageNet, ResNet-50						
	$p \times q$	Batch size	TA (%)	RA (%)	AA (%)	C (s)	T (s)
AT	1×6	512	62.70	40.38	37.46	NA	6022
DAT-PGD w/o LALR	6×6	6×512	57.09	34.02	30.98	865	1932
DAT-PGD	6×6	6×512	63.75	38.45	36.04	898	1960
Fast AT	1×6	512	58.99	40.78	37.18	NA	1544
DAT-FGSM w/o LALR	6×6	6×512	55.04	35.03	32.16	863	1080
DAT-FGSM	6×6	6×512	58.02	40.27	36.02	859	1109

Training setting. Unless specified otherwise, we choose the training perturbation size $\epsilon = 8/255$ and $2/255$ for CIFAR and ImageNet respectively, where recall that ϵ was defined in (1). We also choose 10 steps and 4 steps for PGD attack generation in DAT (and its variants) under CIFAR and ImageNet, respectively. The number of training epochs is given by 100 for CIFAR-10 and 30 for ImageNet. Such training settings are consistent with previous state-of-the-art [Zhang et al., 2019a, Wong et al., 2020]. To implement DAT-FGSM, we find that the use of cyclic learning rate suggested by Fast AT [Wong et al., 2020] becomes over-sensitive to the increase of batch size; see Appendix 4.2. Thus, we adopt the standard piecewise decay step size and an early-stop strategy [Rice et al., 2020] in DAT.

Evaluation setting. Unless specified otherwise, we report robust test accuracy (**RA**) of a learned model against PGD attacks [Madry et al., 2017]. Unless specified otherwise, we choose the perturbation size same as the training ϵ in evaluation, and the number of PGD steps is selected as 20 and 10 for CIFAR and ImageNet, respectively. We will also measure RA against AutoAttacks and the resulting robust accuracy is named **AA**. Further, we measure the standard test accuracy (**TA**) of a model against normal examples. All experiments are run 3 times with different random seeds, and the mean metrics are reported. We consider three different communication protocols, Ring-AllReduce (with one-sided quantization), parameter-server (with double quantization), and high performance computing (HPC) setting (without quantization). To measure the communication time, we use TORCH.DISTRIBUTED package with gloo and nccl as communication backend[1]. We then measure the time of required worker-server communications per epoch.

6.2 EXPERIMENT RESULTS

Adversarial training on ImageNet. In Table 1, we show an overall performance comparison on ImageNet between our proposed DAT variants and baselines in TA, RA, communication and computation efficiency. Note that AT and Fast AT are centralized training baselines using the same number of epochs as distributed training. **First**, we observe that the direct extension from AT (or Fast AT) to its distributed counterpart (namely, DAT-PGD w/o LALR or DAT-FGSM w/o LALR) leads to a large degradation of both RA and TA. **Second**, DAT-PGD (or DAT-FGSM) is able to achieve competitive performance to AT (or Fast AT) and enables a graceful training speedup. In practice, DAT is not able to achieve linear speed-up mainly because of the communication cost. For example, when comparing the computation time of DAT-PGD (batch size 6×512) with that of AT (batch size 512), the computation speed-up (by excluding the communication cost) is given by $(6022)/(1960 - 898) = 5.67$, consistent with the ideal computation gain using $6\times$ larger batch size in DAT-PGD. **Third**, when comparing DAT-FGSM with DAT-PGD, we observe that the former leads to a larger loss in standard

[1] https://pytorch.org/docs/stable/distributed.html

accuracy (around 5%). Moreover, similar to Fast AT, we noted a larger RA variance of DAT-FGSM (around 1.5%) than DAT-PGD (around 0.5%). Thus, the FGSM-based training is less stable than the AT-based one, consistent with [Li et al., 2020].

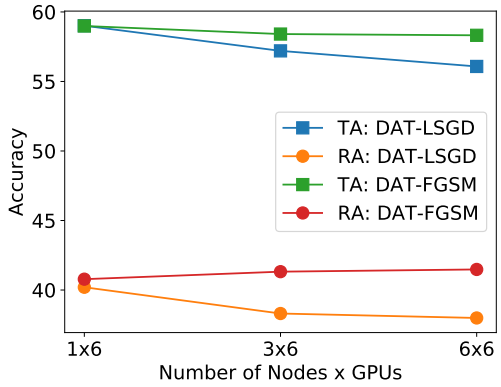
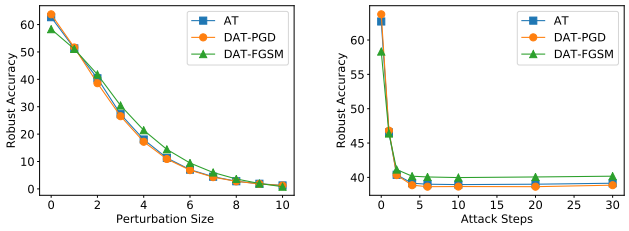


Figure 2: TA/RA comparison between DAT-FGSM and DAT-LSGD vs. node-GPU configurations on (ImageNet, ResNet-50).



(a) (b)

Figure 3: RA against PGD attacks for model trained by DAT-PGD, DAT-FGSM, and AT following (ImageNet, ResNet-50) in Table 1. (a) RA versus different perturbation sizes (over the divisor 255). (b) RA versus different steps.

In Figure 2, we further compare our proposed DAT with the DAT-LSGD baseline [Xie et al., 2019] in terms of TA/RA versus the number of computing nodes. Clearly, our approach scales more gracefully than the baseline, without losing much performance as the batch sizes increases along with the number of computing nodes. It is worth noting that our DAT setup is more challenging than [Xie et al., 2019], which used 128 GPUs but the per-GPU utilization is the 32 batch size. By contrast, although we only use 36 GPUs, the per-GPU batch size is 85. The use of a larger batch size per GPU makes distributed robust training useful when having access to a limited computing budget. Besides, Xie et al. [2019] used the PGD attack generation with a quite large number of attack steps (30) at the training time. This makes the computation time dominated over the node-wise communication time. However, we used a less number of PGD attack steps. In this scenario, the communication time cannot be neglected and prevents the practical distributed implementation from achieving the linear speed-up. For example, when comparing the computation time of DAT-PGD with that of AT in Table 1, the computation speed-up (by excluding communication cost) is given by $6022/(1960-898) = 5.67$, close to the linear rate ($6\times$).

In Figure 3, we evaluate the performance of DAT against PGD attacks of different steps and perturbation sizes (i.e., values of ϵ). We observe that DAT matches robust accuracies of standard AT even against PGD attacks at different values of ϵ and steps.

Adversarially trained smooth classifier. DAT also provides us an effective way to speed up the smooth adversarial training (Smooth-AT) [Salman et al., 2020] for certified robustness. Different from AT, Smooth-AT augments a single training sample with *multiple* Gaussian noisy copies so as to train a Gaussian smoothing-aware classifier. Thus, Smooth-AT requires $N\times$ data batch size and storage capacity in contrast to AT, where N is the number of noisy copies per sample. In the centralized training regime, N can only be set by a small value, e.g., $N = 1$ or 2. However, DAT is able to scale up Smooth-AT with a large value of N, e.g., $N = 20$ in Table 2.

Table 2: Certified accuracy (%) of smooth classifiers trained by Smooth-DAT on (CIFAR-10, ResNet-18) versus ℓ_2 radii. Here smooth classifiers are achieved at two Gaussian noise variance levels, $\sigma = 0.12$ and $\sigma = 0.25$, following [Salman et al., 2020]. And Smooth-AT is implemented using the baseline approach with $N = 2$ and the DAT approach with $N = 20$, respectively.

Method	Smooth classifier ($\sigma = 0.12$)						
	$r = 0.05$	$r = 0.1$	$r = 0.15$	$r = 0.2$	$r = 0.3$	$r = 0.4$	$r = 0.5$
Baseline ($N = 2$)	0.832	0.804	0.762	0.728	0.654	0.545	0
DAT ($N = 20$)	0.838	0.812	0.784	0.748	0.661	0.550	0
Method	Smooth classifier ($\sigma = 0.25$)						
	$r = 0.05$	$r = 0.1$	$r = 0.15$	$r = 0.2$	$r = 0.3$	$r = 0.4$	$r = 0.5$
Baseline ($N = 2$)	0.752	0.730	0.708	0.678	0.625	0.562	0.498
DAT ($N = 20$)	0.764	0.748	0.716	0.688	0.632	0.566	0.514

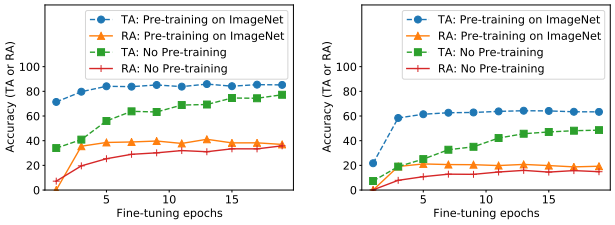
Smooth-AT can produce a provably robust classifier [Cohen et al., 2019]. To be more specific, let $f(\mathbf{x})$ denote a classifier (with input $\mathbf{x}$) trained by Smooth-AT. Then its Gaussian smoothing version, given by $f_{\text{smooth}}(\mathbf{x}) := \arg\max_c \mathbb{P}_{\boldsymbol{\delta}\in\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}[f(\mathbf{x}+\boldsymbol{\delta}) = c]$, can achieve certified robustness, where c is a class label, $\boldsymbol{\delta} \in \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ denotes the standard Gaussian noise with variance σ^2, and $\mathbb{P}$ signifies the majority vote-based prediction probability over multiple noisy samples. The resulting smooth classifier f_{smooth} can then be evaluated at certified accuracy [Cohen et al., 2019], a provable robust guarantee at a given ℓ_2 perturbation radius r.

In Table 2, we present the certified accuracy (CA) of a σ-specified smooth classifier, obtained by either the conventional Smooth-AT approach (baseline using $N = 2$) or the DAT-enabled Smooth-AT method (DAT using $N = 20$).

And we evaluated CA at different ℓ_2-radii. As we can see, DAT yields improved certified robustness over the conventional Smooth-AT with $N = 2$. This demonstrates the advantage of DAT in training provably robust classifiers: The use of a large number of Gaussian noisy samples becomes feasible through distributed training. We also observe that CA drops if the perturbation ℓ_2-radius r increases. This is not surprising since CA is derived by sanity checking if the certified ℓ_2 perturbation radius of a smooth classifier can cover a given r. Besides, the smoother classifier constructed using Gaussian noises of large variance σ tend to be more robust against a larger ℓ_2 perturbation radius, but may hamper the accuracy against perturbations of small ℓ_2 radius. This is consistent with [Cohen et al., 2019] and reveals the tradeoff between accuracy and certified robustness for a σ-specific smooth classifier.

Table 3: DAT with semi-supervision using ResNet-18 or Wide ResNet-28-10 under CIFAR-10 + 500K unlabeled Tiny Images.

Method	ResNet-18, batch size 12×2048				
	TA (%)	RA (%)	AA (%)	C(s)	T(s)
DAT-PGD	87.00	47.34	45.23	86	451
DAT-FGSM	88.00	45.84	43.19	86	124
Method	Wide ResNet-28-10, batch size 12×128				
	TA (%)	RA (%)	AA (%)	C(s)	T(s)
DAT-PGD	89.37	62.06	58.35	302	1020
DAT-FGSM	89.52	61.24	57.65	302	674



(a) Finetuning over CIFAR-10 (b) Finetuning over CIFAR-100

Figure 4: Fine-tuning ResNet-50 (pre-trained on ImageNet) under CIFAR-10 (a) and CIFAR-100 (b). For compassion, adversarial training on CIFAR datasets from scratch (no pretrain) is also presented. Here DAT-PGD is used for both pre-training and fine-tuning at 6 computing nodes.

DAT under unlabeled data In Table 3, we report TA and RA of DAT in the semi-supervised setting [Carmon et al., 2019] with the use of 500K unlabeled images mined from Tiny Images [Carmon et al., 2019]. As we can see, both DAT-PGD and DAT-FGSM scale well even if a $12\times$ batch size is used across 12 machines, each of which has a single GPU. We also compare DAT-PGD with [Carmon et al., 2019] following the latter's architecture, Wide ResNet 28-10. We note that DAT-PGD yields 89.37% TA and 58.35% AA. This is close to the reported 89.69% TA and 59.53% AA in RobustBench [Croce et al., 2020] built upon small-batch adversarial training. Although the use of large data batch may cause a performance loss due to the reduced number of training iterations, the use of data augmentation serves as a remedy for such loss.

DAT from pre-training to fine-tuning. In Figure 4, we investigate if a DAT pre-trained model (ResNet-50) over a source dataset (ImageNet) can offer a fast fine-tuning to a down-stream target dataset (CIFAR-10/100). Here we up-sample a CIFAR image to the same dimension of an ImageNet image before feeding it into the pre-trained model [Shafahi et al., 2020]. Compared with the direct application of DAT to the target dataset (without pre-training), the pre-training enables a fast adaption to the down-stream CIFAR task in both TA and RA within just 3 epochs. Thus, the scalability of DAT to large datasets and multiple nodes offers a great potential to rapidly initialize an adversarially robust base model in the 'pre-training + fine-tuning' paradigm.

Quantization effect in various communication protocols In Table 4, we present how DAT is affected by gradient quantization. As we can see, when the number of bits is reduced, the communication cost and the amount of transmitted data are saved, respectively. However, the use of an aggressive gradient quantization introduces a performance loss. For example, compared with the case of using 32 bits, the most aggressive quantization scheme (8-bit 2-sided quantization in Steps 4 and 7 of Algorithm 1) yields an RA drop around 4% and 7% for DAT-PGD and DAT-FGSM, respectively. In particular, DAT-FGSM is more sensitive to the effect of gradient quantization than DAT-PGD. It is worth noting that our main communication configuration used in previous experiments is Ring-AllReduce that calls for 1-sided (rather than 2-sided) quantization. We further show that if a high performance computing (HPC) cluster of nodes (with NVLink high-speed GPU interconnect [Foley and Danskin, 2017]) is used, the communication cost can be further reduced without causing performance loss.

7 CONCLUSIONS

We proposed distributed adversarial training (DAT) to scale up the training of adversarially robust DNNs over multiple machines. We showed that DAT is general in that it enables large-batch min-max optimization and supports gradient compression and different learning regimes. We proved that under mild conditions, DAT is guaranteed to converge to a first-order stationary point with a sub-linear rate. Empirically, we provided comprehensive experiment results to demonstrate the effectiveness and the usefulness of DAT in training robust DNNs with large datasets and multiple machines. In the future, it will be worthwhile to examine the speedup achieved by DAT in the extreme training cases, e.g., using a significantly large number of attack steps and

Table 4: Effect of gradient quantization on the performance of DAT for various numbers of bits. The training and evaluation settings on (ImageNet, ResNet-50) are consistent with Table 1. The new performance metric 'Data trans. (MB)' represents data transmitted per iteration in the unit MB.

Method	**ImageNet, ResNet-50**				
	# bits	TA (%)	RA (%)	C (s)	Data trans. (MB)
DAT-PGD	32	63.75	38.45	898	2924
DAT-PGD	16	61.77	38.40	850	1462
DAT-PGD	8	56.53	37.90	592	731
DAT-PGD	8 (2-sided)	53.09	34.59	1091	244
DAT-PGD (HPC)	32	63.43	38.55	15	1074
DAT-FGSM	32	58.02	40.27	859	2924
DAT-FGSM	16	54.71	39.29	849	1462
DAT-FGSM	8	50.11	36.38	594	731
DAT-FGSM	8 (2-sided)	48.27	33.20	1013	244
DAT-FGSM (HPC)	32	57.60	41.70	15	310

distributed machines, and extremely large models.

Acknowledgements

Y. Zhang and S. Liu are supported by the Cisco Research grant CG# 70614511. P. Khanduri and M. Hong are supported in part by the NSF grants 1910385 and 1727757. We also thank Dr. Cho-Jui Hsieh for the helpful discussion on early ideas of this paper.

References

D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, 2017.

M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

M. Andriushchenko and N. Flammarion. Understanding and improving fast adversarial training. In *NeurIPS*, 2020.

A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. signSGD: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

A. Boopathy, S. Liu, G. Zhang, P.-Y. Chen, S. Chang, and L. Daniel. Visual interpretability alone helps adversarial robustness, 2020.

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 11190–11201, 2019.

C.-Y. Chen, J. Ni, S. Lu, X. Cui, P.-Y. Chen, X. Sun, N. Wang, S. Venkataramani, V. Srinivasan, W. Zhang, and K. Gopalakrishnan. Scalecom: Scalable sparsified gradient compression for communication-efficient distributed training. In *Advances in Neural Information Processing Systems*, 2020.

X. Chen, S. Liu, R. Sun, and M. Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. In *ICLR*, 2018.

J. Cohen, E. Rosenfeld, and Z. Kolter. Certified adversarial robustness via randomized smoothing. In *ICLR*, 2019.

F. Croce, M. Andriushchenko, V. Sehwag, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.

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

K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song. Physical adversarial examples for object detectors. In *Prof. of the 12th USENIX Workshop on Offensive Technologies (WOOT 18)*, 2018.

D. Foley and J. Danskin. Ultra-performance pascal GPU and NVLink interconnect. *IEEE Micro*, 37(2):7–17, 2017.

R. Gao, T. Cai, H. Li, C.-J. Hsieh, L. Wang, and J. D. Lee. Convergence of adversarial training in overparametrized neural networks. In *Advances in Neural Information Processing Systems*, pages 13029–13040, 2019.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *2015 ICLR*, arXiv preprint arXiv:1412.6572, 2015.

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

K. Hanada, R. Morita, T. Wada, and Y. Fujisaki. Simple synchronous and asynchronous algorithms for distributed minimax optimization. *SICE Journal of Control, Measurement, and System Integration*, 10(6):557–562, 2017.

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, 2016a.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016b.

D. Kang, Y. Sun, D. Hendrycks, T. Brown, and J. Steinhardt. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016*, 2019.

H. Kannan, A. Kurakin, and I. Goodfellow. Adversarial logit pairing, 2018.

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

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

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

B. Li, S. Wang, S. Jana, and L. Carin. Towards understanding fast adversarial training. *arXiv preprint arXiv:2006.03089*, 2020.

M. Liu, Y. Mroueh, W. Zhang, X. Cui, T. Yang, and P. Das. Decentralized parallel algorithm for training generative adversarial nets. *arXiv preprint arXiv:1910.12999*, 2019a.

W. Liu, A. Mokhtari, A. Ozdaglar, S. Pattathil, Z. Shen, and N. Zheng. A decentralized proximal point-type method for saddle point problems. *arXiv preprint arXiv:1910.14380*, 2019b.

S. Lu, M. Razaviyayn, B. Yang, K. Huang, and M. Hong. Finding second-order stationary points efficiently in smooth nonconvex linearly constrained optimization problems. In *NeurIPS*, 2020.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard. Robustness via curvature regularization, and vice versa. In *CVPR*, 2019.

I. Notarnicola, M. Franceschelli, and G. Notarstefano. A duality-based approach for distributed min–max optimization. *IEEE Transactions on Automatic Control*, 64(6): 2559–2566, 2018.

N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli. Adversarial robustness through local linearization. In *NeurIPS*, 2019.

S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *ICLR*, 2018.

L. Rice, E. Wong, and J. Z. Kolter. Overfitting in adversarially robust deep learning. *arXiv preprint arXiv:2002.11569*, 2020.

A. S. Ross and F. Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *AAAI*, 2018.

H. Salman, G. Yang, J. Li, P. Zhang, H. Zhang, I. Razenshteyn, and S. Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. In *NeurIPS*, 2020.

A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! In *NeurIPS*, 2019.

A. Shafahi, P. Saadatpanah, C. Zhu, A. Ghiasi, C. Studer, D. Jacobs, and T. Goldstein. Adversarially robust transfer learning. In *ICLR*, 2020.

A. Sinha, H. Namkoong, and J. Duchi. Certifiable distributional robustness with principled adversarial training. In *ICLR*, 2018.

K. Srivastava, A. Nedić, and D. Stipanović. Distributed min-max optimization in networks. In *Proc. of the 17th International Conference on Digital Signal Processing (DSP)*, 2011.

R. Stanforth, A. Fawzi, P. Kohli, et al. Are labels required for improving adversarial robustness? *arXiv preprint arXiv:1905.13725*, 2019.

S. U. Stich, J.-B. Cordonnier, and M. Jaggi. Sparsified sgd with memory. In *NeurIPS*, 2018.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

I. Tsaknakis, M. Hong, and S. Liu. Decentralized min-max optimization: Formulations, algorithms and applications in network poisoning attack. In *ICASSP*, 2020.

J. Wang, V. Tantia, N. Ballas, and M. Rabbat. Slowmo: Improving communication-efficient distributed sgd with slow momentum. *arXiv preprint arXiv:1910.00643*, 2019a.

Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu. On the convergence and robustness of adversarial training. In *ICML*, pages 6586–6595, 2019b.

J. Wangni, J. Wang, J. Liu, and T. Zhang. Gradient sparsification for communication-efficient distributed optimization. In *NeurIPS*, 2018.

E. Wong and J. Z. Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He. Feature denoising for improving adversarial robustness. In *CVPR*, 2019.

H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *arXiv preprint arXiv:1909.08072*, 2019a.

K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin. Evading real-time person detectors by adversarial t-shirt. *arXiv preprint arXiv:1910.11099*, 2019b.

Y. You, I. Gitman, and B. Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017a.

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

Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer. Imagenet training in minutes. In *Proc. of the 47th International Conference on Parallel Processing*. ACM, 2018.

Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *ICLR*, 2019.

Y. Yu, J. Wu, and L. Huang. Double quantization for communication-efficient distributed optimization. In *NeurIPS*, 2019.

D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong. You only propagate once: Accelerating adversarial training via maximal principle. *arXiv preprint arXiv:1905.00877*, 2019a.

H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. Theoretically principled trade-off between robustness and accuracy. *ICLR*, 2019b.

SUPPLEMENTARY MATERIALS

1 DAT ALGORITHM FRAMEWORK

Algorithm A1 Distributed adversarial training (DAT) for solving problem (2)

1: Initial $\boldsymbol{\theta}_1$, dataset $\mathcal{D}^{(i)}$ for each of M workers, and T iterations
2: **for** Iteration $t = 1, 2, \ldots, T$ **do**
3: **for** Worker $i = 1, 2, \ldots, M$ **do** ▷ Worker
4: Draw a finite-size data batch $\mathcal{B}_t^i \subseteq \mathcal{D}^{(i)}$
5: For each data sample $\mathbf{x} \in \mathcal{B}_t^i$, call for an *inner maximization oracle*:

$$\boldsymbol{\delta}_t^{(i)}(\mathbf{x}) := \underset{\|\boldsymbol{\delta}\|_\infty \leq \epsilon}{\arg\max}\ \phi(\boldsymbol{\theta}_t, \boldsymbol{\delta}; \mathbf{x}), \tag{A1}$$

where we omit the label or possible pseudo-label y of $\mathbf{x}$ for brevity
6: Computing local gradient of f_i in (2) with respect to $\boldsymbol{\theta}$ given perturbed samples:

$$\mathbf{g}_t^{(i)} = \lambda \mathbb{E}_{\mathbf{x} \in \mathcal{B}_t^{(i)}}[\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}_t; \mathbf{x})] + \mathbb{E}_{\mathbf{x} \in \mathcal{B}_t^{(i)}}[\nabla_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}_t; \mathbf{x} + \boldsymbol{\delta}_t^{(i)}(\mathbf{x}))] \tag{A2}$$

7: (*Optional*) Call for *gradient quantizer* $Q(\cdot)$ and transmit $Q(\mathbf{g}_t^{(i)})$ to server
8: **end for**
9: Gradient aggregation at server: ▷ Server

$$\hat{\mathbf{g}}_t = \frac{1}{M} \sum_{i=1}^{M} Q(\mathbf{g}_t^{(i)}) \tag{A3}$$

10: (*Optional*) Call for *gradient quantizer* $\hat{\mathbf{g}}_t \leftarrow Q(\hat{\mathbf{g}}_t)$, and transmit $\hat{\mathbf{g}}_t$ to workers:
11: **for** Worker $i = 1, 2, \ldots, M$ **do** ▷ Worker
12: Call for an *outer minimization oracle* $\mathcal{A}(\cdot)$ to update $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_{t+1} = \mathcal{A}(\boldsymbol{\theta}_t \hat{\mathbf{g}}_t, \eta_t), \qquad \eta_t \text{ is learning rate} \tag{A4}$$

13: **end for**
14: **end for**

Additional details on gradient quantization Let b denote the number of bits ($b \leq 32$), and thus there exists $s = 2^b$ quantization levels. We specify the gradient quantization operation $Q(\cdot)$ in Algorithm A1 as the *randomized quantizer* [Alistarh et al., 2017, Yu et al., 2019]. Formally, the quantization operation at the ith coordinate of a vector $\mathbf{g}$ is given by [Alistarh et al., 2017]

$$Q(g_i) = \|\mathbf{g}\|_2 \cdot \text{sign}(g_i) \cdot \xi_i(g_i, s), \quad \forall i \in \{1, 2, \ldots, d\}. \tag{A5}$$

In (A5), $\xi_i(g_i, s)$ is a random number drawn as follows. Given $|g_i|/\|\mathbf{g}\|_2 \in [l/s, (l+1)/s]$ for some $l \in \mathbb{N}^+$ and $0 \leq l < s$, we then have

$$\xi_i(g_i, s) = \begin{cases} l/s & \text{with probability } 1 - (s|g_i|/\|\mathbf{g}\|_2 - l) \\ (l+1)/s & \text{with probability } (s|g_i|/\|\mathbf{g}\|_2 - l), \end{cases} \tag{A6}$$

where $|a|$ denotes the absolute value of a scalar a, and $\|\mathbf{a}\|_2$ denotes the ℓ_2 norm of a vector $\mathbf{a}$. The rationale behind using (A5) is that $Q(g_i)$ is an *unbiased* estimate of g_i, namely, $\mathbb{E}_{\xi_i(g_i,s)}[Q(g_i)] = g_i$, with bounded variance. Moreover, we at most need $(32 + d + bd)$ bits to transmit the quantized $Q(\mathbf{g})$, where 32 bits for $\|\mathbf{g}\|_2$, 1 bit for sign of g_i and b bits for $\xi_i(g_i, s)$, whereas it needs $32d$ bits for a single-precision $\mathbf{g}$. Clearly, a small b saves the communication cost. We note that if every worker performs as a server in DAT, then the quantization operation at Step 10 of Algorithm A1 is no longer needed. In this case, the communication network becomes fully connected. With synchronized communication, this is favored for training DNNs under the All-reduce operation.

2 THEORETICAL RESULTS

In this section, we will quantify the convergence behaviour of the proposed DAT algorithm. First, we define the following notations:

$$\Phi_i(\boldsymbol{\theta}, \mathbf{x}) = \max_{\|\boldsymbol{\delta}^{(i)}\|_\infty \leq \epsilon} \phi(\boldsymbol{\theta}, \boldsymbol{\delta}^{(i)}; \mathbf{x}), \quad \text{and} \quad \Phi_i(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \in \mathcal{D}^{(i)}} \Phi_i(\boldsymbol{\theta}; \mathbf{x}). \tag{A7}$$

We also define

$$l_i(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \in \mathcal{D}^{(i)}} l(\boldsymbol{\theta}; \mathbf{x}), \tag{A8}$$

where the label y of $\mathbf{x}$ is omitted for labeled data. Then, the objective function of problem (2) can be expressed in the compact way

$$\Psi(\boldsymbol{\theta}) = \frac{1}{M} \sum_{i=1}^{M} \lambda l_i(\boldsymbol{\theta}) + \Phi_i(\boldsymbol{\theta}) \tag{A9}$$

and the optimization problem is then given by $\min_{\boldsymbol{\theta}} \Psi(\boldsymbol{\theta})$.

Therefore, it is clear that if a point $\boldsymbol{\theta}^\star$ satisfies

$$\|\nabla_{\boldsymbol{\theta}} \Psi(\boldsymbol{\theta}^\star)\| \leq \xi, \tag{A10}$$

then we say $\boldsymbol{\theta}^\star$ is a ξ approximate first-order stationary point (FOSP) of problem (2).

Prior to delving into the convergence analysis of DAT, we make the following assumptions.

2.1 ASSUMPTIONS

A1. Assume objective function has layer-wise Lipschitz continuous gradients with constant L_i for each layer

$$\|\nabla_i \Psi(\boldsymbol{\theta}_{\cdot,i}) - \nabla_i \Psi(\boldsymbol{\theta}'_{\cdot,i})\| \leq L_i \|\boldsymbol{\theta}_{\cdot,i} - \boldsymbol{\theta}'_{\cdot,i}\|, \forall i \in [h]. \tag{A11}$$

where $\nabla_i \Psi(\boldsymbol{\theta}_{\cdot,i})$ denotes the gradient w.r.t. the variables at the ith layer. Also, we assume that $\Psi(\boldsymbol{\theta})$ is lower bounded, i.e., $\Psi^\star := \min_{\boldsymbol{\theta}} \Psi(\boldsymbol{\theta}) > -\infty$ and bounded gradient estimate, i.e., $\|\nabla \hat{\mathbf{g}}_t^{(i)}\| \leq G$.

A2. Assume that $\phi(\boldsymbol{\theta}, \boldsymbol{\delta}; \mathbf{x})$ is strongly concave with respect to $\boldsymbol{\delta}$ with parameter μ and has the following gradient Lipschitz continuity with constant L_ϕ:

$$\|\nabla_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}, \boldsymbol{\delta}; \mathbf{x}) - \nabla_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}, \boldsymbol{\delta}'; \mathbf{x})\| \leq L_\phi \|\boldsymbol{\delta} - \boldsymbol{\delta}'\|. \tag{A12}$$

A3. Assume that the gradient estimate is unbiased and has bounded variance, i.e.,

$$\mathbb{E}_{\mathbf{x} \in \mathcal{B}^{(i)}} [\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \mathbf{x})] = \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}), \forall i, \tag{A13}$$

$$\mathbb{E}_{\mathbf{x} \in \mathcal{B}^{(i)}} [\nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}; \mathbf{x})] = \nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}), \forall i, \tag{A14}$$

where recall that $\mathcal{B}^{(i)}$ denotes a data batch used at worker i, $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) := \frac{1}{M} \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} l_i(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}) := \frac{1}{M} \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} \Phi_i(\boldsymbol{\theta})$; and

$$\mathbb{E}_{\mathbf{x} \in \mathcal{B}^{(i)}} \|\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \mathbf{x}) - \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta})\|^2 \leq \sigma^2, \forall i \tag{A15}$$

$$\mathbb{E}_{\mathbf{x} \in \mathcal{B}^{(i)}} \|\nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}; \mathbf{x}) - \nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta})\|^2 \leq \sigma^2, \forall i. \tag{A16}$$

Further, we define a component-wise bounded variance of the gradient estimate

$$\mathbb{E}_{\mathbf{x} \in \mathcal{B}^{(i)}} \|[\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \mathbf{x})]_{jk} - [\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta})]_{jk}\|^2 \leq \sigma_{jk}^2, \forall i, \tag{A17}$$

$$\mathbb{E}_{\mathbf{x} \in \mathcal{B}^{(i)}} \|[\nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}; \mathbf{x})]_{jk} - [\nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta})]_{jk}\|^2 \leq \sigma_{jk}'^2, \forall i, \tag{A18}$$

where j denotes the index of the layer, and k denotes the index of entry at each layer. Under A3, we have $\sum_{j=1}^{h} \sum_{k=1}^{d_j} \max\{\sigma_{jk}^2, \sigma_{jk}'^2\} \leq \sigma^2$

A4. Assume that the component wise compression error has bounded variance

$$\mathbb{E}[(Q([\mathbf{g}^{(i)}(\boldsymbol{\theta})]_{jk}) - [\mathbf{g}^{(i)}(\boldsymbol{\theta})]_{jk})^2] \leq \delta_{jk}^2, \forall i. \tag{A19}$$

The assumption A4 is satisfied as the randomized quantization is used [Alistarh et al., 2017, Lemma 3.1].

2.2 ORACLE OF MAXIMIZATION

In practice, $\Phi_i(\boldsymbol{\theta};\mathbf{x}), \forall i$ may not be obtained, since the inner loop needs to iterate by the infinite number of iterations to achieve the exact maximum point. Therefore, we allow some numerical error term resulted in the maximization step at (A1). This consideration makes the convergence analysis more realistic.

First, we have the following criterion to measure the closeness of the approximate maximizer to the optimal one.

Definition 1. *Under A2, if point $\boldsymbol{\delta}(\mathbf{x})$ satisfies*

$$\max_{\boldsymbol{\delta}\leq\|\epsilon\|} \langle \boldsymbol{\delta}-\boldsymbol{\delta}^*(\mathbf{x}), \nabla_{\boldsymbol{\delta}}\phi(\boldsymbol{\theta},\boldsymbol{\delta}^*(\mathbf{x});\mathbf{x})\rangle \leq \varepsilon \tag{A20}$$

then, it is a ε approximate solution to $\boldsymbol{\delta}^*(\mathbf{x})$, where

$$\boldsymbol{\delta}^*(\mathbf{x}) := \arg\max_{\boldsymbol{\delta}\leq\|\epsilon\|} \phi(\boldsymbol{\theta},\boldsymbol{\delta};\mathbf{x}). \tag{A21}$$

and $\mathbf{x}$ denotes the sampled data.

Condition (A20) is standard for defining approximate solutions of an optimization problem over a compact feasible set and has been widely studied in [Wang et al., 2019b, Lu et al., 2020].

In the following, we can show that when the inner maximization problem is solved accurately enough, the gradients of function $\phi(\boldsymbol{\theta},\boldsymbol{\delta}(\mathbf{x});\mathbf{x})$ at $\boldsymbol{\delta}(\mathbf{x})$ and $\boldsymbol{\delta}^*(\mathbf{x})$ are also close. A similar claim of this fact has been shown in [Wang et al., 2019b, Lemma 2]. For completeness of the analysis, we provide the specific statement for our problem here and give the detailed proof as well.

Lemma 1. *Let $\boldsymbol{\delta}_t^{(k)}$ be the $(\mu\varepsilon)/L_\phi^2$ approximate solution of the inner maximization problem for worker k, i.e., $\max_{\boldsymbol{\delta}^{(k)}} \phi(\boldsymbol{\theta},\boldsymbol{\delta}^{(k)};\mathbf{x}_t)$, where $\mathbf{x}_t$ denotes the sampled data at the tth iteration of DAT. Under A2, we have*

$$\left\|\nabla_{\boldsymbol{\theta}}\phi\left(\boldsymbol{\theta}_t,\boldsymbol{\delta}_t^{(k)}(\mathbf{x}_t);\mathbf{x}_t\right)-\nabla_{\boldsymbol{\theta}}\phi\left(\boldsymbol{\theta}_t,(\boldsymbol{\delta}^*)_t^{(k)}(\mathbf{x}_t);\mathbf{x}_t\right)\right\|^2 \leq \varepsilon. \tag{A22}$$

Throughout the convergence analysis, we assume that $\boldsymbol{\delta}_t^{(k)}(\mathbf{x}_t), \forall k,t$ are all the $(\mu\varepsilon)/L_\phi^2$ approximate solutions of the inner maximization problem. Let us define

$$\left\|[\nabla\phi(\boldsymbol{\theta}_t,\boldsymbol{\delta}_t^{(k)}(\mathbf{x}_t);\mathbf{x}_t)]_{ij}-[\nabla\phi(\boldsymbol{\theta}_t,(\boldsymbol{\delta}^*)_t^{(k)}(\mathbf{x}_t);\mathbf{x}_t]_{ij}\right\|^2 = \varepsilon_{ij}. \tag{A23}$$

From Lemma 1, we know that when $\boldsymbol{\delta}_t^{(k)}(\mathbf{x}_t)$ is a $(\mu\varepsilon)/L_\phi^2$ approximate solution, then

$$\sum_{i=1}^{h}\sum_{j=1}^{d_i}\varepsilon_{ij} = \sum_{i=1}^{h}\sum_{j=1}^{d_i}\left\|[\nabla\phi(\boldsymbol{\theta}_t,\boldsymbol{\delta}_t^{(k)}(\mathbf{x}_t);\mathbf{x}_t)]_{ij}-[\nabla\phi(\boldsymbol{\theta}_t,(\boldsymbol{\delta}^*)_t^{(k)}(\mathbf{x}_t);\mathbf{x}_t]_{ij}\right\|^2 \leq \varepsilon. \tag{A24}$$

2.3 FORMAL STATEMENTS OF CONVERGENCE RATE GUARANTEES

In what follows, we provide the formal statement of convergence rate of DAT. In our analysis, we focus on the 1-sided quantization, namely, Step 10 of Algorithm A1 is omitted, and specify the outer minimization oracle by LAMB [You et al., 2019], see Algorithm A2. The addition and multiplication operations in LAMB are component-wise.

Theorem 2. *Under A1-A4, suppose that $\{\boldsymbol{\theta}_t\}$ is generated by DAT for a total number of T iterations, and let the problem dimension at each layer be $d_i = d/h$. Then the convergence rate of DAT is given by*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla_{\boldsymbol{\theta}}\Psi(\boldsymbol{\theta}_t)\|^2 \leq \frac{\Delta_\Psi}{\eta_t c_l C T} + 2\left(\varepsilon+\frac{(1+\lambda)\sigma^2}{MB}\right) + 4\delta^2 + \frac{\kappa\sqrt{3}}{C}\|\boldsymbol{\chi}\|_1 + \frac{\eta_t c_u \kappa \|L\|_1}{2C}. \tag{A25}$$

where $\Delta_\Psi := \mathbb{E}[\Psi(\boldsymbol{\theta}_1)] - \Psi^\star$, η_t is the learning rate, $\kappa = c_u/c_l$, c_l and c_u are constants used in LALR (3), $\boldsymbol{\chi}$ is an error term with the $(ih+j)$th entry being $\sqrt{\frac{(1+\lambda)\sigma_{ij}^2}{MB}+\varepsilon_{ij}+\delta_{ij}^2}$, ε and ε_{ij} were given in (A24), $L = [L_1,\ldots,L_h]^T$, $C = \frac{1}{4}\sqrt{\frac{h(1-\beta_2)}{G^2 d}}$, $0<\beta_2<1$ is given in LAMB, $B = \min\{|\mathcal{B}^{(i)}|, \forall i\}$, and G is given in A1.

Remark 1. When the batch size is large, i.e., $B \sim \sqrt{T}$, then the gradient estimate error will be $\mathcal{O}(\sigma^2/\sqrt{T})$. Further, it is worth noting that different from the convergence results of LAMB, there is a linear speedup of deceasing the gradient estimate error in DAT with respect to M, i.e., $\mathcal{O}(\sigma^2/(M\sqrt{T}))$, which is the advantage of using multiple computing nodes.

Remark 2. Note that A4 implies $\mathbb{E}[(Q([\mathbf{g}^{(k)}(\boldsymbol{\theta})]_{ij}) - [\mathbf{g}^{(k)}(\boldsymbol{\theta})]_{ij}\|^2] \leq \sum_{i=1}^{h}\sum_{j=1}^{d_i}\delta_{ij}^2 := \delta^2$. From [Alistarh et al., 2017, Lemma 3.1], we know that $\delta^2 \leq \min\{d/s^2, \sqrt{d}/s\}G^2$. Recall that $s = 2^b$, where b is the number of quantization bits.

Therefore, with a proper choice of the parameters, we can have the following convergence result that has been shown in Theorem 1.

Corollary 1. *Under the same conditions of Theorem 2, if we choose*

$$\eta_t \sim \mathcal{O}(1/\sqrt{T}), \quad \varepsilon \sim \mathcal{O}(\xi^2), \tag{A26}$$

we then have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla_{\boldsymbol{\theta}}\Psi(\boldsymbol{\theta}_t)\|^2 \leq \frac{\Delta_\Psi}{c_l C\sqrt{T}} + \frac{(1+\lambda)\sigma^2}{MB} + \frac{c_u\kappa\|L\|_1}{2C\sqrt{T}} + \mathcal{O}\left(\xi, \frac{\sigma}{\sqrt{MT}}, \min\left\{\frac{d}{4^b}, \frac{\sqrt{d}}{2^b}\right\}\right). \tag{A27}$$

In summary, when the batch size is large enough, DAT converges to a first-order stationary point of problem (2) and there is a linear speed-up in terms of M with respect to σ^2. Next, we provide the details of the proof.

3 PROOF DETAILS

3.1 PRELIMINARIES

In the proof, we use the following inequality and notations.

1. Young's inequality with parameter ϵ is

$$\langle \mathbf{x}, \mathbf{y} \rangle \leq \frac{1}{2\epsilon}\|\mathbf{x}\|^2 + \frac{\epsilon}{2}\|\mathbf{y}\|^2, \tag{A28}$$

where $\mathbf{x}, \mathbf{y}$ are two vectors.

2. Define the historical trajectory of the iterates as $\mathcal{F}_t = \{\boldsymbol{\theta}_{t-1}, \ldots, \boldsymbol{\theta}_1\}$.

3. We denote vector $[\mathbf{x}]_i$ as the parameters at the ith layer of the neural net and $[\mathbf{x}]_{ij}$ represents the jth entry of the parameter at the ith layer.

4. We define

$$\mathbf{g}_t := \frac{1}{M}\sum_{i=1}^{M} \mathbb{E}_{\mathbf{x}_t \in \mathcal{B}^{(i)}} \left(\lambda \nabla l(\boldsymbol{\theta}_t; \mathbf{x}_t) + \nabla_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}_t, \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) \right) = \frac{1}{M}\sum_{i=1}^{M} \mathbf{g}_t^{(i)}. \tag{A29}$$

3.2 DETAILS OF LAMB ALGORITHM

Algorithm A2 LAMB [You et al., 2019]

Input: learning rate η_t, $0 < \beta_1, \beta_2 < 1$, scaling function $\tau(\cdot)$, $\zeta > 0$
for $t = 1, \ldots$ **do**
  $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\hat{\mathbf{g}}_t$, where $\hat{\mathbf{g}}_t$ is given by (A3)
  $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\hat{\mathbf{g}}_t^2$
  $\mathbf{m}_t = \mathbf{m}_t/(1-\beta_1^t)$
  $\mathbf{v}_t = \mathbf{v}_t/(1-\beta_2^t)$
  Compute ratio $\mathbf{u}_t = \frac{\mathbf{m}_t}{\sqrt{\mathbf{v}_t}+\zeta}$
end for
Update

$$\boldsymbol{\theta}_{t+1,i} = \boldsymbol{\theta}_{t,i} - \frac{\eta_t \tau(\|\boldsymbol{\theta}_{t,i}\|)}{\|\mathbf{u}_{t,i}\|}\mathbf{u}_{t,i}. \tag{A30}$$

3.3 PROOF OF LEMMA 1

Proof. From A2, we have

$$\left\| \nabla \phi\left(\boldsymbol{\theta}_t, \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t\right) - \nabla \phi\left(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t\right) \right\| \leq L_\phi \|\boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t) - (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t)\|. \tag{A31}$$

Also, we know that function $\phi(\boldsymbol{\theta}, \boldsymbol{\delta}, \mathbf{x})$ is strongly concave with respect to $\boldsymbol{\delta}$, so we have

$$\begin{aligned} &\mu \|\boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t) - (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t)\| \\ &\qquad \leq \left\langle \nabla_{\boldsymbol{\delta}} \phi(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) - \nabla_{\boldsymbol{\delta}} \phi(\boldsymbol{\theta}_t, \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t), \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t) - (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t) \right\rangle. \end{aligned} \tag{A32}$$

Next, we have two conditions about the qualities of solutions $\boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t)$ and $(\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t)$. First, we know that $\boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t)$ is a-ε approximate solution to $(\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t)$, so we have

$$\left\langle (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t) - \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t), \nabla_{\boldsymbol{\delta}} \phi(\boldsymbol{\theta}_t, \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) \right\rangle \leq \varepsilon. \tag{A33}$$

Second, since $(\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t)$ is the optimal solution, it satisfies

$$\left\langle (\boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t) - (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t), \nabla_{\boldsymbol{\delta}}\phi(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) \right\rangle \leq 0. \tag{A34}$$

Adding them together, we can obtain

$$\left\langle \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t) - (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t), \nabla_{\boldsymbol{\delta}}\phi(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) - \nabla_{\boldsymbol{\delta}}\phi(\boldsymbol{\theta}_t, \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) \right\rangle \leq \varepsilon. \tag{A35}$$

Substituting (A35) into (A32), we can get

$$\mu \|\boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t) - (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t)\|^2 \leq \varepsilon. \tag{A36}$$

Combining (A31), we have

$$\left\| \nabla\phi(\boldsymbol{\theta}_t, \boldsymbol{\delta}_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) - \nabla\phi(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) \right\|^2 \leq L_\phi^2 \frac{\varepsilon}{\mu}. \tag{A37}$$

□

3.4 DESCENT OF QUANTIZED LAMB

First, we provide the following lemma as a stepping stone for the subsequent analysis.

Lemma 2. *Under A1–A3, suppose that sequence $\{\boldsymbol{\theta}_t\}$ is generated by DAT. Then, we have*

$$\mathbb{E}[-\langle \nabla\Psi(\boldsymbol{\theta}_t), \hat{\mathbf{g}}_t \rangle] \leq -\frac{\mathbb{E}\|\nabla\Psi(\boldsymbol{\theta}_t)\|^2}{2} + \varepsilon + \frac{(1+\lambda)\sigma^2}{MB}. \tag{A38}$$

Proof. From (A21), (A7) and A2, we know that

$$\nabla_{\boldsymbol{\theta}}\Phi_i(\boldsymbol{\theta}, \mathbf{x}) = \nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}, (\boldsymbol{\delta}^*)^{(i)}(\mathbf{x}); \mathbf{x}), \tag{A39}$$

so we can get

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\Psi(\boldsymbol{\theta}) =& \frac{1}{M}\sum_{i=1}^{M} \lambda\nabla_{\boldsymbol{\theta}} l_i(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\Phi_i(\boldsymbol{\theta}) && \text{(A40)}\\
=& \lambda\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) + \frac{1}{M}\sum_{i=1}^{M} \mathbb{E}_{\mathbf{x}\in\mathcal{D}^{(i)}} \nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}, (\boldsymbol{\delta}^*)^{(i)}(\mathbf{x}); \mathbf{x}) && \text{(A41)}\\
:=& \bar{\mathbf{g}}(\boldsymbol{\theta}). && \text{(A42)}
\end{aligned}$$

Then, we have

$$\begin{aligned}
\mathbb{E}\langle \nabla\Psi(\boldsymbol{\theta}_t), \mathbf{g}_t \rangle =& \mathbb{E}\langle \nabla\Psi(\boldsymbol{\theta}_t), \bar{\mathbf{g}}_t \rangle + \mathbb{E}\langle \nabla\Psi(\boldsymbol{\theta}_t), \mathbf{g}_t - \bar{\mathbf{g}}_t \rangle && \text{(A43)}\\
=& \mathbb{E}_{\mathcal{F}_t}\mathbb{E}_{\mathbf{x}_t|\mathcal{F}_t}\langle \nabla\Psi(\boldsymbol{\theta}_t), \bar{\mathbf{g}}_t \rangle + \mathbb{E}\langle \nabla\Psi(\boldsymbol{\theta}_t), \mathbf{g}_t - \bar{\mathbf{g}}_t \rangle && \text{(A44)}\\
\overset{(A42)}{=}& \mathbb{E}\|\nabla\Psi(\boldsymbol{\theta}_t)\|^2 + \mathbb{E}\langle \nabla\Psi(\boldsymbol{\theta}_t), \mathbf{g}_t - \bar{\mathbf{g}}_t \rangle && \text{(A45)}\\
=& \mathbb{E}\|\nabla\Psi(\boldsymbol{\theta}_t)\|^2 + \mathbb{E}\langle \nabla\Psi(\boldsymbol{\theta}_t), \mathbf{g}_t - \mathbf{g}_t^* \rangle + \mathbb{E}\langle \nabla\Psi(\boldsymbol{\theta}_t), \mathbf{g}_t^* - \bar{\mathbf{g}}_t \rangle && \text{(A46)}
\end{aligned}$$

where

$$\bar{\mathbf{g}}_t := \frac{1}{M}\sum_{i=1}^{M} \mathbb{E}_{\mathbf{x}_t\in\mathcal{D}^{(i)}} \left(\lambda\nabla l(\boldsymbol{\theta}_t, \mathbf{x}_t) + \nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) \right) = \lambda\nabla l(\boldsymbol{\theta}_t) + \nabla\Phi(\boldsymbol{\theta}_t), \tag{A47}$$

and

$$\mathbf{g}_t^* := \frac{1}{M}\sum_{i=1}^{M} \mathbb{E}_{\mathbf{x}_t\in\mathcal{B}^{(i)}} \left(\lambda\nabla l(\boldsymbol{\theta}_t, \mathbf{x}_t) + \nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)_t^{(i)}(\mathbf{x}_t); \mathbf{x}_t) \right). \tag{A48}$$

Next, we can quantify the different between $\mathbf{g}_t$ and $\mathbf{g}_t^*$ by gradient Lipschitz continuity of function $\tau(\cdot)$ as the following

$$\mathbb{E}\|\mathbf{g}_t - \mathbf{g}_t^*\|^2 \overset{(a)}{\leq} \frac{1}{M}\sum_{i=1}^{M} \mathbb{E}_{\mathcal{F}_t}\mathbb{E}_{\mathbf{x}_t|\mathcal{F}_t}\left[\|\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)^{(i)}(\mathbf{x}_t); \mathbf{x}_t) - \nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}_t, \boldsymbol{\delta}^{(i)}(\mathbf{x}_t); \mathbf{x}_t)\|^2\right] \overset{(A24)}{\leq} \varepsilon \tag{A49}$$

where in (a) we use Jensen's inequality.

And the difference between $\bar{\mathbf{g}}_t$ and $\mathbf{g}_t^*$ can be upper bounded by

$$\begin{aligned}
\mathbb{E}\|\bar{\mathbf{g}}_t - \mathbf{g}_t^*\|^2 =& \mathbb{E}_{\mathcal{F}_t}\left\|\frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{\mathbf{x}_t|\mathcal{F}_t}\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}_t, (\boldsymbol{\delta}^*)^{(i)}(\mathbf{x}_t); \mathbf{x}_t) - \nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}_t)\right\|^2 \\
&+ \lambda\mathbb{E}_{\mathcal{F}_t}\left\|\frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{\mathbf{x}_t|\mathcal{F}_t}\nabla l(\boldsymbol{\theta}_t; \mathbf{x}_t) - \nabla l(\boldsymbol{\theta}_t)\right\|^2 && \text{(A50)} \\
\overset{A3}{=}& \frac{(1+\lambda)\sigma^2}{MB}. && \text{(A51)}
\end{aligned}$$

Applying Young's inequality with parameter 2, we have

$$\begin{aligned}
\mathbb{E}[-\langle\nabla\Psi(\boldsymbol{\theta}_t), \mathbf{g}_t\rangle] \leq& -\mathbb{E}\|\nabla\Psi(\boldsymbol{\theta}_t)\|^2 + \frac{\mathbb{E}\|\nabla\Psi(\boldsymbol{\theta}_t)\|^2}{2} + \mathbb{E}\|\bar{\mathbf{g}}_t - \mathbf{g}_t^*\|^2 + \mathbb{E}\|\mathbf{g}_t^* - \mathbf{g}_t\|^2 && \text{(A52)} \\
\overset{(A49)}{\leq}& -\frac{\mathbb{E}\|\nabla\Psi(\boldsymbol{\theta}_t)\|^2}{2} + \varepsilon + \frac{(1+\lambda)\sigma^2}{MB}. && \text{(A53)}
\end{aligned}$$

□

3.5 PROOF OF THEOREM 2

Proof. We set $\beta_1 = 0$ in LAMB for simplicity. From gradient Lipschitz continuity, we have

$$\begin{aligned}
\Psi(\boldsymbol{\theta}_{t+1}) \overset{A1}{\leq}& \Psi(\boldsymbol{\theta}_t) + \sum_{i=1}^{h}\langle[\nabla_{\boldsymbol{\theta}}\Psi(\boldsymbol{\theta}_t)]_i, \boldsymbol{\theta}_{t+1,i} - \boldsymbol{\theta}_{t,i}\rangle + \sum_{i=1}^{h}\frac{L_i}{2}\|\boldsymbol{\theta}_{t+1,i} - \boldsymbol{\theta}_{t,i}\|^2 && \text{(A54)} \\
\overset{(a)}{\leq}& \Psi(\boldsymbol{\theta}_t) \underbrace{-\eta_t\sum_{i=1}^{h}\sum_{j=1}^{d_i}\tau(\|\boldsymbol{\theta}_{t,i}\|)\left\langle[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}, \frac{[\mathbf{u}_t]_{ij}}{\|\mathbf{u}_{t,i}\|}\right\rangle}_{:=\boldsymbol{\mathcal{R}}} + \sum_{i=1}^{h}\frac{\eta_t^2 c_u^2 L_i}{2}, && \text{(A55)}
\end{aligned}$$

where in (a) we use (A30), and the upper bound of $\tau(\|\boldsymbol{\theta}_{t,i}\|)$.

Next, we split term R as two parts by leveraging $\text{sign}([\nabla\Psi(\boldsymbol{\theta}_t)]_{ij})$ and $\text{sign}([\mathbf{u}_t]_{ij})$ as follows.

$$\begin{aligned}
\boldsymbol{\mathcal{R}} = &- \eta_t \sum_{i=1}^{h}\sum_{j=1}^{d_i} \tau(\|\boldsymbol{\theta}_{t,i}\|)[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij} \frac{[\mathbf{u}_t]_{ij}}{\|\mathbf{u}_{t,i}\|} \mathbb{1}\left(\text{sign}([\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}) = \text{sign}([\mathbf{u}_t]_{ij})\right) \\
&- \eta_t \sum_{i=1}^{h}\sum_{j=1}^{d_i} \tau(\|\boldsymbol{\theta}_{t,i}\|)[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij} \frac{[\mathbf{u}_t]_{ij}}{\|\mathbf{u}_{t,i}\|} \mathbb{1}\left(\text{sign}([\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}) \neq \text{sign}([\mathbf{u}_t]_{ij})\right) \qquad \text{(A56)} \\
\overset{(a)}{\leq} &- \eta_t c_l \sum_{i=1}^{h}\sum_{j=1}^{d_i} \sqrt{\frac{1-\beta_2}{G^2 d_i}}[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}[\hat{\mathbf{g}}_t]_{ij} \mathbb{1}\left(\text{sign}([\nabla[\Psi(\boldsymbol{\theta}_t)]_{ij}) = \text{sign}([\hat{\mathbf{g}}_t]_{ij})\right) \\
&- \eta_t \sum_{i=1}^{h}\sum_{j=1}^{d_i} \tau(\|\boldsymbol{\theta}_{t,i}\|)[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij} \frac{[\mathbf{u}_t]_{ij}}{\|\mathbf{u}_{t,i}\|} \mathbb{1}\left(\text{sign}([\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}) \neq \text{sign}([\mathbf{u}_t]_{ij})\right) \qquad \text{(A57)} \\
\overset{(b)}{\leq} &- \eta_t c_l \sum_{i=1}^{h}\sum_{j=1}^{d_i} \sqrt{\frac{1-\beta_2}{G^2 d_i}}[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}[\hat{\mathbf{g}}_t]_{ij} \\
&- \eta_t \sum_{i=1}^{h}\sum_{j=1}^{d_i} \tau(\|\boldsymbol{\theta}_{t,i}\|)[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij} \frac{[\mathbf{u}_t]_{ij}}{\|\mathbf{u}_{t,i}\|} \mathbb{1}\left(\text{sign}([\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}) \neq \text{sign}([\mathbf{u}_t]_{ij})\right). \qquad \text{(A58)}
\end{aligned}$$

where in (a) we use the fact that $\|\mathbf{u}_{t,i}\| \leq \sqrt{\frac{d_i}{1-\beta_2}}$ and $\sqrt{\mathbf{v}_t} \leq G$, and in (b) we add

$$-\eta_t c_l \sum_{i=1}^{h}\sum_{j=1}^{d_i} \sqrt{\frac{1-\beta_2}{G^2 d_i}}[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}[\hat{\mathbf{g}}_t]_{ij} \mathbb{1}\left(\text{sign}([\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}) \neq \text{sign}([\hat{\mathbf{g}}_t]_{ij})\right) \geq 0. \qquad \text{(A59)}$$

Taking expectation on both sides of (A58), we have the following:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\mathcal{R}}] \leq &\underbrace{-\eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \sum_{i=1}^{h}\sum_{j=1}^{d_i} \mathbb{E}[[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}[\hat{\mathbf{g}}_t]_{ij}}_{:=\boldsymbol{\mathcal{U}}} \\
&\underbrace{+ \eta_t c_u \sum_{i=1}^{h}\sum_{j=1}^{d_i} \mathbb{E}\left[[\nabla\Psi(\boldsymbol{\theta}_t)]_{ij} \mathbb{1}\left(\text{sign}([\nabla\Psi(\boldsymbol{\theta}_t)]_{ij}) \neq \text{sign}([\mathbf{u}_t]_{ij})\right)\right]}_{:=\boldsymbol{\mathcal{V}}}. \qquad \text{(A60)}
\end{aligned}$$

Next, we will get the upper bounds of $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ separably as follows. First, we write the inner product between $[\nabla\Psi(\boldsymbol{\theta})]_{ij}$ and $[\hat{\mathbf{g}}_t]_{ij}$ more compactly,

$$\begin{aligned}
\boldsymbol{\mathcal{U}} \leq &- \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \sum_{i=1}^{h} \mathbb{E}\left\langle [\nabla\Psi(\boldsymbol{\theta})]_i, [\hat{\mathbf{g}}_t]_i \right\rangle \qquad \text{(A61)} \\
\leq &- \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \sum_{i=1}^{h} \mathbb{E}\left\langle [\nabla\Psi(\boldsymbol{\theta}_t)]_i, [\hat{\mathbf{g}}_t]_i - [\mathbf{g}_t]_i + [\mathbf{g}_t]_i \right\rangle \qquad \text{(A62)} \\
\leq &- \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \left(\mathbb{E}\left\langle \nabla\Psi(\boldsymbol{\theta}), \mathbf{g}_t \right\rangle + \sum_{i=1}^{h} \mathbb{E}\left\langle [\nabla\Psi(\boldsymbol{\theta}_t)]_i, [\hat{\mathbf{g}}_t]_i - [\mathbf{g}_t]_i \right\rangle \right). \qquad \text{(A63)}
\end{aligned}$$

Applying Lemma 2, we can get

$$\begin{aligned}
\boldsymbol{\mathcal{U}} \overset{(A38)}{\leq} & -\eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \frac{1}{2} \mathbb{E}\|\nabla \Psi(\boldsymbol{\theta}_t)\|^2 + \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \left(\varepsilon + \frac{(1+\lambda)\sigma^2}{MB}\right) \\
& - \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \sum_{i=1}^{h} \mathbb{E} \left\langle [\nabla \Psi(\boldsymbol{\theta}_t)]_i, [\hat{\mathbf{g}}_t]_i - [\mathbf{g}_t]_i \right\rangle \qquad \text{(A64)} \\
\overset{(a)}{\leq} & -\eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \frac{1}{2} \mathbb{E}\|\nabla \Psi(\boldsymbol{\theta}_t)\|^2 + \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \left(\varepsilon + \frac{(1+\lambda)\sigma^2}{MB}\right) \\
& + \frac{\eta_t c_l}{4} \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \mathbb{E}\|\nabla \Psi(\boldsymbol{\theta}_t)\|^2 + c_l \eta_t \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \mathbb{E}\|\hat{\mathbf{g}}_t - \mathbf{g}_t\|^2 \qquad \text{(A65)} \\
\overset{(b)}{\leq} & -\frac{\eta_t c_l}{4} \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \frac{1}{2} \mathbb{E}\|\nabla \Psi(\boldsymbol{\theta}_t)\|^2 + \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \left(\varepsilon + \frac{(1+\lambda)\sigma^2}{MB}\right) \\
& + \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \delta^2 \qquad \text{(A66)}
\end{aligned}$$

where we use the in (a) we use Young's inequality (with parameter 2), and in (b) we have

$$\mathbb{E}\|\hat{\mathbf{g}}_t - \mathbf{g}_t\|^2 = \mathbb{E}\left\| \frac{1}{M} \sum_{i=1}^{M} Q(\mathbf{g}_t^{(i)}) - \mathbf{g}_t^{(i)} \right\|^2 \overset{A4}{\leq} \delta^2. \qquad \text{(A67)}$$

Second, we give the upper of $\boldsymbol{\mathcal{V}}$:

$$\boldsymbol{\mathcal{V}} \leq \eta_t c_u \sum_{i=1}^{h} \sum_{j=1}^{d_i} [\nabla \Psi(\boldsymbol{\theta}_t)]_{ij} \underbrace{\mathbb{P}\left(\text{sign}([\nabla \Psi(\boldsymbol{\theta}_t)]_{ij}) \neq \text{sign}([\hat{\mathbf{g}}_t]_{ij})\right)}_{:=\boldsymbol{\mathcal{W}}} \qquad \text{(A68)}$$

where the upper bound of $\boldsymbol{\mathcal{W}}$ can be quantified by using Markov's inequality followed by Jensen's inequality as the following:

$$\begin{aligned}
\boldsymbol{\mathcal{W}} =& \mathbb{P}\left(\text{sign}([\nabla \Psi(\boldsymbol{\theta}_t)]_{ij}) \neq \text{sign}([\hat{\mathbf{g}}_t]_{ij})\right) \\
\leq& \mathbb{P}[|[\nabla \Psi(\boldsymbol{\theta}_t)]_{ij} - [\hat{\mathbf{g}}_t]_{ij}| > [\nabla \Psi(\boldsymbol{\theta}_t)]_{ij}] \qquad \text{(A69)} \\
\leq& \frac{\mathbb{E}[[\nabla \Psi(\boldsymbol{\theta}_t)]_{ij} - [\hat{\mathbf{g}}_t]_{ij}]}{|[\nabla \Psi(\boldsymbol{\theta}_t)]_{ij}|} \qquad \text{(A70)} \\
\leq& \frac{\sqrt{\mathbb{E}[([\nabla \Psi(\boldsymbol{\theta}_t)]_{ij} - [\hat{\mathbf{g}}_t]_{ij})^2]}}{|[\nabla \Psi(\boldsymbol{\theta}_t)]_{ij}|} \qquad \text{(A71)} \\
\overset{(A42)}{\leq}& \frac{\sqrt{\mathbb{E}[([\bar{\mathbf{g}}_t]_{ij} - [\mathbf{g}_t^*]_{ij} + [\mathbf{g}_t^*]_{ij} - [\mathbf{g}_t]_{ij} + [\mathbf{g}_t]_{ij} - [\hat{\mathbf{g}}_t]_{ij})^2]}}{|[\nabla \Psi(\boldsymbol{\theta}_t)]_{ij}|} \qquad \text{(A72)} \\
\overset{(a)}{\leq}& \sqrt{3} \frac{\sqrt{\frac{(1+\lambda)\sigma_{ij}^2}{M|\mathcal{B}|} + \epsilon_{ij} + \delta_{ij}^2}}{|[\nabla \Psi(\boldsymbol{\theta}_t)]_{ij}|} \qquad \text{(A73)}
\end{aligned}$$

where (a) is true due to the following relations: *i*) from (A51), we have

$$\mathbb{E}[([\bar{\mathbf{g}}_t]_{ij} - [\mathbf{g}_t^*]_{ij})^2] \leq \frac{(1+\lambda)\sigma_{ij}^2}{MB}; \qquad \text{(A74)}$$

ii) from (A49), we can get

$$\mathbb{E}[([\mathbf{g}_t]_{ij} - [\mathbf{g}_t^*]_{ij})^2] \leq \varepsilon_{ij}; \qquad \text{(A75)}$$

and *iii*) from (A67), we know

$$\mathbb{E}[([\hat{\mathbf{g}}_t]_{ij} - [\mathbf{g}_t]_{ij})^2] \leq \delta_{ij}^2. \qquad \text{(A76)}$$

Therefore, combining (A55) with the upper bound of $\boldsymbol{\mathcal{U}}$ shown in (A66) and $\boldsymbol{\mathcal{V}}$ shown in (A68)(A73), we have

$$\begin{aligned}\mathbb{E}[\Psi(\boldsymbol{\theta}_{t+1})] \leq & \mathbb{E}[\Psi(\boldsymbol{\theta}_t)] - \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \frac{1}{4} \mathbb{E}\|\nabla \Psi(\boldsymbol{\theta}_t)\|^2 + \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \left(\varepsilon + \frac{(1+\lambda)\sigma^2}{MB}\right) \\ & + \eta_t c_l \sqrt{\frac{h(1-\beta_2)}{G^2 d}} \delta^2 + \eta_t c_u \sqrt{3} \sum_{i=1}^{h} \sum_{j=1}^{d_i} \sqrt{\frac{(1+\lambda)\sigma_{ij}^2}{MB} + \varepsilon_{ij} + \delta_{ij}^2} + \frac{\eta_t^2 c_u^2 \sum_{i=1}^{h} L_i}{2}. \end{aligned} \tag{A77}$$

Note that the error vector $\boldsymbol{\chi}$ is defined as the following

$$\boldsymbol{\chi} = \begin{bmatrix} \sqrt{\frac{(1+\lambda)\sigma_{11}^2}{M|\mathcal{B}|} + \varepsilon_{11} + \delta_{11}^2} \\ \vdots \\ \sqrt{\frac{(1+\lambda)\sigma_{ij}^2}{M|\mathcal{B}|} + \varepsilon_{ij} + \delta_{ij}^2} \\ \vdots \\ \sqrt{\frac{(1+\lambda)\sigma_{hd_h}^2}{M|\mathcal{B}|} + \varepsilon_{hd_h} + \delta_{hd_h}^2} \end{bmatrix} \in \mathbb{R}^d, \tag{A78}$$

and we have

$$L = \begin{bmatrix} L_1 \\ \vdots \\ L_h \end{bmatrix} \in \mathbb{R}^h. \tag{A79}$$

Recall

$$\kappa = \frac{c_u}{c_l}. \tag{A80}$$

Rearranging the terms, we can arrive at

$$\begin{aligned}\underbrace{\sqrt{\frac{h(1-\beta_2)}{G^2 d}} \frac{1}{4}}_{:=C} \left(\|\nabla \Psi(\boldsymbol{\theta}_t)\|^2\right) \leq & \frac{\mathbb{E}[\Psi(\boldsymbol{\theta}_t)] - \mathbb{E}[\Psi(\boldsymbol{\theta}_{t+1})]}{\eta_t c_l} + 4C\delta^2 + 2C\left(\varepsilon + \frac{(1+\lambda)\sigma^2}{MB}\right) \\ & + \sqrt{3}\kappa \|\boldsymbol{\chi}\|_1 + \frac{\eta_t c_u \kappa \|L\|_1}{2}. \end{aligned} \tag{A81}$$

Applying the telescoping sum over $t = 1, \ldots, T$, we have

$$\begin{aligned}\frac{1}{T} \sum_{t=1}^{\top} \mathbb{E}\|\nabla_{\boldsymbol{\theta}} \Psi(\boldsymbol{\theta}_t)\|^2 \leq & \frac{\mathbb{E}[\Psi(\boldsymbol{\theta}_1)] - \mathbb{E}[\Psi(\boldsymbol{\theta}_{T+1})]}{\eta_t c_l C T} + 2\left(\varepsilon + \frac{(1+\lambda)\sigma^2}{MB}\right) + 4\delta^2 \\ & + \frac{\kappa\sqrt{3}}{C} \|\boldsymbol{\chi}\|_1 + \frac{\eta_t c_u \kappa \|L\|_1}{2C}. \end{aligned} \tag{A82}$$

□

4 ADDITIONAL EXPERIMENTS

4.1 TRAINING DETAILS

ImageNet AT and Fast AT experiments are conducted at a single computing node with dual 22-core CPU, 512GB RAM and 6 Nvidia V100 GPUs. The training epoch is 30 by calling for the momentum SGD optimizer. The weight decay and momentum parameters are set to 0.0001 and 0.9. The initial learning rate is set to 0.1 (tuned over $\{0.01, 0.05, 0.1, 0.2\}$), which is decayed by $\times 1/10$ at the training epoch $20, 25, 28$, respectively.

ImageNet DAT experiments are conducted at $\{1, 3, 6\}$ computing nodes with dual 22-core CPU, 512GB RAM and 6 Nvidia V100 GPUs. The training epoch is 30 by calling for the LAMB optimizer. The weight decay is set to 0.0001. β_1 and β_2 are set to 0.9 and 0.999. The initial learning rate η_1 is tuned over $\{0.01, 0.05, 0.1, 0.2, 0.4\}$, which is decayed by $\times 1/10$ at the training epoch $20, 25, 28$, respectively. To execute algorithms with the initial learning rate η_1 greater than 0.2, we choose the model weights after 5-epoch warm-up as its initialization for DAT, where each warm-up epoch k uses the linearly increased learning rate $(k/5)\eta_1$.

4.2 ADDITIONAL RESULTS

Discussion on cyclic learning rate. It was shown in [Wong et al., 2020] that the use of a cyclic learning rate (CLR) trick can further accelerate the Fast AT algorithm in the small-batch setting [Wong et al., 2020]. In Figure A1, we present the performance of Fast AT with CLR versus batch sizes. We observe that when CLR meets the large-batch setting, it becomes significantly worse than its performance in the small-batch setting. The reason is that CLR requires a certain number of iterations to proceed with the cyclic schedule. However, the use of large data batch only results in a small amount of iterations by fixing the number of epochs.

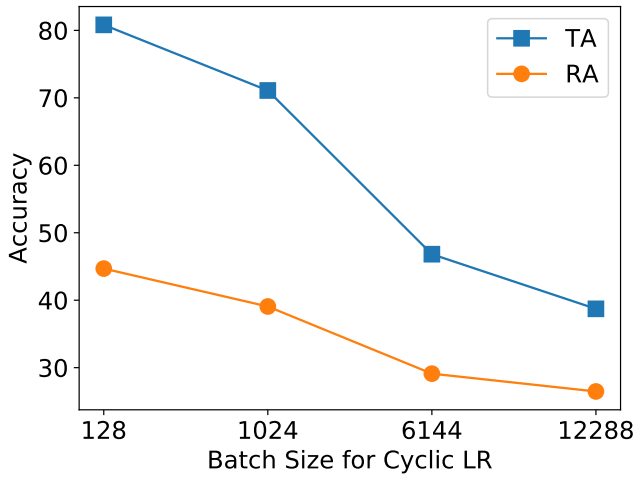


Figure A1: TA/RA of Fast AT with CLR versus batch sizes on (CIFAR-10, ResNet-18).

Additional details on HPC setups. To further reduce communication cost, we also conduct DAT at a HPC cluster. The computing nodes of the cluster are connected with InfiniBand (IB) and PCIe Gen4 switch. To compare with results in Table 1, we use 6 of 57 nodes of the cluster. Each node has 6 Nvidia V100s which are interconnected with NVLink. We use Nvida NCCL as communication backend. In Table 4, we have presented the performance of DAT for ImageNet, ResNet-50 with use of HPC compared to standard (non-HPC) distributed system.