Communication-Efficient Distributed Deep Learning via Federated Dynamic Averaging

Michail Theologitis
Technical University of Crete
mtheologitis@tuc.gr

Georgios Frangias
Technical University of Crete
gfrangias@tuc.gr

Georgios Anestis
Technical University of Crete
ganestis@tuc.gr

Vasilis Samoladas
Technical University of Crete
vsamoladas@tuc.gr

Antonios Deligiannakis
Technical University of Crete
adeli@softnet.tuc.gr

ABSTRACT

Driven by the ever-growing volume and decentralized nature of data, coupled with the escalating size of modern models, distributed deep learning (DDL) has been entrenched as the preferred paradigm for training. However, frequent synchronization of DL models, encompassing millions to many billions of parameters, creates a communication bottleneck, severely hindering scalability. Worse yet, DDL algorithms typically waste valuable bandwidth, and make themselves less practical in bandwidth-constrained federated settings, by relying on overly simplistic, periodic, and rigid synchronization schedules. To address these shortcomings, we propose Federated Dynamic Averaging (FDA), a communication-efficient DDL strategy that dynamically triggers synchronization based on the value of the model variance. Through extensive experiments across a wide range of learning tasks we demonstrate that FDA reduces communication cost by orders of magnitude, compared to both traditional and cutting-edge communication-efficient algorithms. Remarkably, FDA achieves this without sacrificing convergence speed — in stark contrast to the trade-offs encountered in the field. Additionally, we show that FDA maintains robust performance across diverse data heterogeneity settings.

1 INTRODUCTION

Deep Learning (DL) stands at the forefront of the most significant technological advancements in recent times, with Deep Neural Networks (DNNs) achieving remarkable success in fields such as computer vision [34], natural language processing [7, 16] and speech recognition [5]. These rapid advancements coincide with the big data era, marked by an unprecedented scale of training datasets [54, 64]. These datasets are not only growing in size but are often physically distributed and cannot be easily centralized due to business considerations, privacy concerns, bandwidth limitations, and data sovereignty laws [30, 35, 44]. In response, Distributed Deep Learning (DDL) has emerged as an alternative paradigm to the traditional centralized approach [9, 49, 81], offering efficient handling of large-scale data across multiple worker-nodes, enhancing the speed of training DL models and paving the way for more scalable and resilient DL applications [1, 38, 47].

Most DDL methods are iterative, where, in each iteration, some amount of local training is followed by *synchronization* of the local model with the global one. The predominant method, based on the bulk synchronous parallel (BSP) approach [69], is to average the local model updates and then apply the average update to each local model [14, 81]. Less synchronized variants have also been proposed, to ameliorate the effect of *straggler workers* [12, 18, 21, 25, 46, 48] but sometimes compromise convergence speed and model quality.

A critical challenge inherent in the traditional techniques, especially in federated DL settings, where models are huge and worker interconnections are slow, is the communication bottleneck, restricting system scalability [57, 67]. This leads to a low computation-to-communication ratio [41, 58]. Addressing the communication overhead to expedite DDL algorithms has been a focal point of research for many years; speeding-up SGD is arguably among the most impactful and transformative problems in machine learning [71].

The most direct method to alleviate the communication burden is to reduce the frequency of communication rounds. Local-SGD is the prime example of this approach. It allows workers to perform τ local update steps on their models before aggregating them, as opposed to averaging the updates in every step [22, 77, 78]. Although Local-SGD is effective in reducing communication while maintaining similar accuracy levels [61, 71], determining the optimal value of τ presents a significant challenge, with only a handful of studies offering theoretical insights into its influence on convergence [62, 71, 77, 79].

To further reduce communication costs of Local-SGD, more sophisticated communication strategies introduce varying sequences of local update steps $\{\tau_0, ..., \tau_R\}$, instead of a fixed τ. In [70], in order to minimize convergence error with respect to wall-time, the authors proposed a decreasing sequence of local update steps. Conversely, the focus in [22] was on reducing the number of communication rounds for a fixed number of model updates and prescribed an increasing sequence. These contrasting approaches underscore the multifaceted nature of communication strategies in distributed deep learning, highlighting not only the absence of a one-size-fits-all solution but also the growing need for dynamic, context-aware strategies that can continuously adapt to the specific intricacies of the learning task.

Main Contributions. Our work addresses critical efficiency challenges in DDL, particularly in communication-constraint environments, such as the ones encountered in Federated Learning (FL) [30] applications. We introduce Federated Dynamic Averaging (FDA), a novel, adaptive distributed deep learning strategy, massively improving communication efficiency over previous work. Our contributions are as follows:

- FDA dynamically decides to synchronize local nodes when *model variance across nodes* exceeds a threshold. This strategy drastically reduces communication requirements, while

preserving cohesive progress towards the shared training objective.

- We evaluate and compare FDA with other DDL algorithms through a comprehensive suite of experiments with diverse datasets, models, and tasks. This extensive approach—requiring over 200K GPU hours—compensates for the stochasticity associated with training DNNs, thereby enabling us to draw more definitive conclusions.
- We demonstrate that FDA outperforms traditional and contemporary FL algorithms by orders of magnitude in communication, while maintaining equivalent model performance. Furthermore, it effectively balances the competing demands of reducing communication and computation, providing greatly improved trade-offs.
- We show that FDA remains robust and efficient under various data distributions (Non-IID settings), maintaining comparable performance to the IID case. Furthermore, we demonstrate that FDA leads to models with excellent generalization, inherently mitigating overfitting exhibited by previous techniques.

2 RELATED WORK

Problem formulation. Consider distributed training of deep neural networks over multiple workers [13, 39]. In this setting, each worker has access to its own set of training data $\mathcal{D}_k$, and the collective goal is to find a common model $\mathbf{w} \in \mathbb{R}^d$ by minimizing the overall training loss. This scenario can be effectively modeled as a distributed optimization problem, formulated as follows:

$$\underset{\mathbf{w} \in \mathbb{R}^d}{\text{minimize}} \quad F(\mathbf{w}) \triangleq \frac{1}{K} \sum_{k=1}^{K} F_k(\mathbf{w}) \tag{1}$$

where K is the number of workers and $F_k(\mathbf{w}) \triangleq \mathbb{E}_{\zeta_k \sim \mathcal{D}_k} [\ell(\mathbf{w}; \zeta_k)]$ is the local objective function for worker k. Function $\ell(\mathbf{w}; \zeta_k)$ represents the *loss* for data sample ζ_k given model w.

Communication efficient Local-SGD. The work in [39] decomposes each round into two phases. In the first phase, each worker runs Local-SGD with $\tau = I_1$, while the second phase run I_2 steps with $\tau = 1$; [39] proposes to exponentially decay I_1 every M rounds. In the heterogeneous setting, the work in [50], by analysing the convergence rate, proposes an increasing sequence of local update steps for strongly-convex local objectives, and suggests fixed local steps for convex, non-convex local objectives. The study in [76] dynamically increases batch sizes to reduce communication rounds, maintaining the same convergence rate as SSP-SGD. However, the large-batch approach leads to poor generalization [26, 40, 56], a challenge addressed by the post-local SGD method [40], which divides training into two phases: BSP-SGD followed by Local-SGD with a fixed number of steps. In the Lazily Aggregated Algorithm (LAG) [8], a different approach was taken, using only new gradients from some selected workers and reusing the outdated gradients from the rest, which essentially skips communication rounds.

Federated Averaging (FedAvg) [44] is another representative communication efficient Local-SGD algorithm, which is a pivotal method in Federated Learning (FL) [30]. In the FL setting with edge

computing systems, the work in [72] tries to find the optimal synchronization period τ subject to local computation and aggregation constraints. Recently [45], in the FL setting with the assumption of strongly-convex objectives, by analysing the balance between fast convergence and higher-round completion rate, a decaying local update step scheme emerged.

Accelerating convergence. An indirect, yet highly effective, way to mitigate the communication burden in DDL, is to speed up convergence. Consequently, recent works have deployed accelerated versions of SGD to the distributed setting. Specifically, FedAdam [52] extends Adam [33] and FedAvgM [27] extends SGD with momentum (SGD-M) [65]. Recently, Mime [31] provides a framework to adapt arbitrary centralized optimization algorithms to the FL setting. However, these methods still suffer from the model divergence problem, particularly in heterogeneous settings. When solving (1), the disparity between each worker's optimal solution $\mathbf{w}_k^*$ for their objective F_k, and the global optimum $\mathbf{w}^*$ for F, can potentially cause worker models to diverge (drift) towards their disparate minima [32, 52]. The result is slow and unstable convergence with significant communication overhead. To address this problem, the SCAFFOLD algorithm [32] used control-variates (in the same spirit to SVRG [29]), with significant speed-up. FedProx [55] re-parametarized FedAvg [44] by adding L^2 regularization in the workers' objectives to be near the global model. Lastly, FedDyn [2] improved upon these ideas with a dynamic regularizer making sure that if local models converge to a consensus, this consensus point aligns with the stationary point of the global objective function.

Compression. To reduce communication overhead in DDL, significant efforts have been directed towards minimizing message sizes. Key strategies include sparsification, where only crucial components of information are transmitted, as explored in the works [3, 4, 63], and quantization techniques, which involve transmitting only quantized gradients, as detailed in studies like [51, 59]. Notably, these compression techniques can be effectively combined with Local-SGD methods to enhance communication-efficiency further. A prime example is Qsparse-local-SGD [6], which integrates aggressive sparsification and quantization with Local-SGD, achieving substantial communication savings to reach specific target accuracies. A comprehensive survey of compression techniques in DDL can be found in [73].

3 FEDERATED DYNAMIC AVERAGING

Notation. At each time instance t, each worker k independently maintains its own vector of model parameters,[1] denoted as $\mathbf{w}_t^{(k)} \in \mathbb{R}^d$. Let $\Delta\mathbf{w}_t^{(k)}$ be the update vector [20] computed by some optimization algorithm (e.g., SGD, Adam) encompassing the learning rate and relevant gradients. In a distributed learning step t, each worker k applies the following update:

$$\mathbf{w}_{t+1}^{(k)} = \mathbf{w}_t^{(k)} + \Delta\mathbf{w}_t^{(k)} \tag{2}$$

Let $\overline{\mathbf{w}}_t = \frac{1}{K} \sum_{k=1}^{K} \mathbf{w}_t^{(k)}$ represent the average model at time t and $\overline{\mathbf{w}}_{t_0}$ the initial model shared among the workers at the start of the current communication round. Moreover, we introduce the *drift*

---

[1]The terms "model" and "model parameters" are used interchangeably, as is common in the literature.

vector $\mathbf{u}_t^{(k)} = \mathbf{w}_t^{(k)} - \overline{\mathbf{w}}_{t_0}$, that is, the change to the local model of the k-th worker at time t since the beginning of the current round at time t_0, and $\overline{\mathbf{u}}_t = \frac{1}{K} \sum_{k=1}^{K} \mathbf{u}_t^{(k)} = \overline{\mathbf{w}}_t - \overline{\mathbf{w}}_{t_0}$ as the average drift.

Model Variance. The *model variance* quantifies the dispersion or spread of the worker models around the average model. At time t, it is defined as:

$$\mathcal{V}ar(\mathbf{w}_t) = \frac{1}{K} \sum_{k=1}^{K} \left\| \mathbf{w}_t^{(k)} - \overline{\mathbf{w}}_t \right\|_2^2 = \frac{1}{K} \sum_{k=1}^{K} \left\| \mathbf{w}_t^{(k)} - \mathbf{w}_{t_0} + \mathbf{w}_{t_0} - \overline{\mathbf{w}}_t \right\|_2^2$$

$$= \frac{1}{K} \sum_{k=1}^{K} \left\| \mathbf{u}_t^{(k)} - \overline{\mathbf{u}}_t \right\|_2^2 = \ldots = \left(\frac{1}{K} \sum_{k=1}^{K} \left\| \mathbf{u}_t^{(k)} \right\|_2^2 \right) - \|\overline{\mathbf{u}}_t\|_2^2 \quad (3)$$

This measure provides insight into how closely aligned the workers' models are at any given time. High variance indicates that the models are widely spread out, essentially drifting apart, leading to a lack of cohesion in the aggregated model. Conversely, a moderate-low variance suggests that the workers' models are closely aligned, working collectively towards the shared objective. The variance plays a critical role in our approach, as it helps us gauge the state of the training process making informed decisions about round termination.

Round Invariant. We introduce the *Round Invariant* (RI), a condition that bounds model variance below a threshold Θ:

$$\mathcal{V}ar(\mathbf{w}_t) \leq \Theta \quad (4)$$

Our algorithm continuously monitors the model variance to maintain the RI. If the variance estimate exceeds the threshold Θ at any point t, the training round immediately terminates. This triggers synchronization, where all local model weights $\mathbf{w}_t^{(k)}$ are averaged and consolidated into the new global model $\overline{\mathbf{w}}_t$.

Monitoring the RI. At this point, our focus shifts to devising a method to monitor the RI, as defined in (4). We begin by defining the *local state* for each worker k, denoted by $S_k(t) \in \mathbb{R}^p$, which encapsulates information from worker k at time t; it is updated arbitrarily. Next, we introduce the *global state*, $S(t) \in \mathbb{R}^p$, which represents the collective state of the distributed system at time t.

$$S(t) = \frac{1}{K} \sum_{k=1}^{K} S_k(t) \quad (5)$$

Lastly, we define a special non-linear function $H : \mathbb{R}^p \to \mathbb{R}$, with the following property:

$$H(S(t)) \leq \Theta \implies \mathcal{V}ar(\mathbf{w}_t) \leq \Theta \quad \forall\, t \quad (6)$$

Conceptually, if we define $S_k(t) = \left(\left\| \mathbf{u}_t^{(k)} \right\|_2^2, \mathbf{u}_t^{(k)} \right) \in \mathbb{R} \times \mathbb{R}^d$ and $H(v, x) = v - \|x\|_2^2$, then, it immediately follows from (3), (5) that $H(S(t)) = \mathcal{V}ar(\mathbf{w}_t)$ and property (6) is satisfied. Direct monitoring of the RI is hindered by the high dimensionality of $S_k(t)$, with d potentially ranging from millions to many billions, values commonly found in DNNs. To mitigate this communication burden, we must apply dimensionality reduction to the local drifts $\mathbf{u}_t^{(k)}$ and identify an appropriate function H, accordingly. However, this reduction in information within local states causes RI monitoring to become approximate. Upcoming sections will detail

---

Algorithm 1 Federated Dynamic Averaging - FDA

Require: $S_k(t)$: The local state $S_k(t) \in \mathbb{R} \times \mathbb{R}^p$ with $p \ll d$
Require: $H(x)$: A function s.t. $H(S(t)) \leq \Theta \implies \mathcal{V}ar(\mathbf{w}_t) \leq \Theta$
Require: K: The number of workers indexed by k
Require: Θ: The model variance threshold
Require: b: The local mini-batch size

1: Initialize $\mathbf{w}_1 \in \mathbb{R}^d$
2: **for** each round $t = 1, 2, \ldots$ **do**
3:     **communicate** $\mathbf{w}_t$ to all workers
4:     **repeat**
5:         **for** each worker $k = 1, \ldots, K$ **in parallel do**
6:             $\mathcal{B}_k \leftarrow$ (sample a batch of size b from $\mathcal{D}_k$)
7:             $\mathbf{w}_t^{(k)} \leftarrow \mathbf{w}_t^{(k)} - \eta \nabla \ell(\mathbf{w}_t^{(k)}; \mathcal{B}_k)$ ▷ In-place
8:         **communicate** $S_k(t)$
9:     **until** $H(S(t)) > \Theta$ ▷ As per (5)
10:     **for** each worker $k = 1, \ldots, K$ **in parallel do**
11:         **communicate** $\mathbf{w}_t^{(k)}$
12:     $\mathbf{w}_{t+1} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \mathbf{w}_t^{(k)}$

---

two strategies–termed "linear", and "sketch"–each offering a different balance between communication efficiency and approximation accuracy.

Algorithm. The proposed FDA algorithm is formalized in Algorithm 1. To streamline notation, t also denotes each training round, which unfolds into three main parts:

(1) **Broadcast**: Model $\mathbf{w}_t$ is distributed to all workers (Line 3).
(2) **Local Training**: As long as we can guarantee the RI, workers continue training concurrently (Lines 4-9). Specifically, workers perform in-parallel updates to their local models $\mathbf{w}_t^{(k)}$ using mini-batch SGD, or some other optimization algorithm, by processing their local data (Lines 5-7). After each update, workers compute and broadcast their local states $S_k(t)$ (Line 8). Then, the global state $S(t)$ is calculated via (5), and using the function H, an estimate of the model variance is derived and compared with the threshold Θ (Line 9). Depending on this comparison, the local training stage either repeats or terminates.
(3) **Model Averaging**: This stage is triggered when we can no longer guarantee the RI, i.e., $H(S(t)) > \Theta$. At this point, all local models are communicated and then averaged to form $\mathbf{w}_{t+1}$, that is, the next round's global model (Lines 10-12).

3.1 Linear Approximation

In the linear approach, we reduce the $\mathbf{u}_t^{(k)}$ vector to a scalar, that is, $\left\langle \xi, \mathbf{u}_t^{(k)} \right\rangle \in \mathbb{R}$ where $\xi \in \mathbb{R}^d$ is any unit vector, and $\langle \cdot, \cdot \rangle$ is the dot product. We define:

$$S_k(t) = \left(\left\| \mathbf{u}_t^{(k)} \right\|_2^2, \left\langle \xi, \mathbf{u}_t^{(k)} \right\rangle \right) \in \mathbb{R} \times \mathbb{R}, \quad \|\xi\|_2 = 1$$

$$H(v, x) = v - x^2$$

Then, the condition $H(S(t)) \le \Theta$ implies $\mathcal{V}ar(\mathbf{w}_t) \le \Theta$, since:

$$H(S(t)) \overset{(5)}{=} H\left(\frac{1}{K}\sum_{k=1}^{K} S_k(t)\right) = \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \left|\frac{1}{K}\sum_{k=1}^{K}\left\langle \xi, \mathbf{u}_t^{(k)}\right\rangle\right|^2$$

$$= \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - |\langle \xi, \overline{\mathbf{u}}_t\rangle|^2 \ge \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \|\overline{\mathbf{u}}_t\|_2^2 = \mathcal{V}ar(\mathbf{w}_t)$$

Furthermore, a random choice of ξ is likely to perform poorly (terminate a round prematurely), as it is likely to be close to orthogonal to $\overline{\mathbf{u}}_t$. A good choice would be a vector correlated to $\overline{\mathbf{u}}_t$. A heuristic choice is to take $\overline{\mathbf{u}}_{t_0}$, i.e., the change vector right before the current round started. All workers can estimate this without communication cost as the difference of the models at the beginning of the current and previous rounds and use $\xi = \frac{w_{t_0} - w_{t_{-1}}}{\|w_{t_0} - w_{t_{-1}}\|_2}$.

3.2 Sketch Approximation

A better estimator for $\|\overline{\mathbf{u}}_t\|_2^2$ can be obtained through the utilization of AMS sketches, as detailed in [11]. An AMS sketch of a vector $\mathbf{v} \in \mathbb{R}^d$ is an $l \times m$ real matrix Ξ:

$$\mathrm{sk}(\mathbf{v}) = \Xi = \begin{bmatrix} \xi_1 & \xi_2 & \dots & \xi_l \end{bmatrix}^\top \in \mathbb{R}^{l \times m} \quad, \quad l \cdot m \ll d$$

The $\mathrm{sk}(\cdot)$ operator is linear and can be computed in $O(l \cdot d)$ steps. Let $l = O(\log 1/\delta)$ and $m = O(1/\epsilon^2)$. The function

$$\mathcal{M}_2(\Xi) = \mathrm{median}\left\{\|\xi_i\|_2^2 \ , \ i = 1, \dots, l\right\}$$

is an excellent estimator of $\|\mathbf{v}\|_2^2$, as with probability at least $(1 - \delta)$, $\mathcal{M}_2(\mathrm{sk}(\mathbf{v}))$ is within ϵ-relative error of $\|\mathbf{v}\|_2^2$. We then define:

$$S_k(t) = \left(\left\|\mathbf{u}_t^{(k)}\right\|_2^2, \mathrm{sk}\left(\mathbf{u}_t^{(k)}\right)\right) \in \mathbb{R} \times \mathbb{R}^{l \times m}$$

$$H(\mathbf{v}, \Xi) = \mathbf{v} - \frac{1}{1 + \epsilon}\mathcal{M}_2(\Xi)$$

Then, the condition $H(S(t)) \le \Theta$ implies $\mathcal{V}ar(\mathbf{w}_t) \le \Theta$ with probability at least $(1 - \delta)$. To prove this, we proceed as follows: we first observe $\mathcal{M}_2\left(\frac{1}{K}\sum_{k=1}^{K}\mathrm{sk}\left(\mathbf{u}_t^{(k)}\right)\right) = \mathcal{M}_2\left(\mathrm{sk}\left(\frac{1}{K}\sum_{k=1}^{K}\mathbf{u}_t^{(k)}\right)\right) = \mathcal{M}_2(\mathrm{sk}(\overline{\mathbf{u}}_t)) \le (1 + \epsilon)\|\overline{\mathbf{u}}_t\|_2^2$ with probability at least $(1 - \delta)$. Then:

$$H(S(t)) = \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \frac{1}{1 + \epsilon}\mathcal{M}_2\left(\frac{1}{K}\sum_{k=1}^{K}\mathrm{sk}\left(\mathbf{u}_t^{(k)}\right)\right)$$

$$\ge \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \|\overline{\mathbf{u}}_t\|_2^2 = \mathcal{V}ar(\mathbf{w}_t) \text{ with prob. at least } (1 - \delta)$$

4 EXPERIMENTS

4.1 Setup

Platform. We employ TensorFlow [1], integrated with Keras [10], as the platform for conducting our experiments. We modified TensorFlow to include our FDA variants and all competitive algorithms.

Datasets & Models. The core experiments involve training Convolutional Neural Networks (CNNs) of varying sizes and complexities on two datasets: MNIST [15] and CIFAR-10 [36]. For the MNIST dataset, we employ LeNet-5 [37], composed of approximately 62 thousand parameters, and a modified version of VGG16 [60], denoted as VGG16*, consisting of 2.6 million parameters. VGG16* was specifically adapted for the MNIST dataset, a less demanding

learning problem compared to ImageNet [54], for which VGG16 was designed. In VGG16*, we omitted the 512-channel convolutional blocks and downscaled the final two fully connected (FC) layers from 4096 to 512 units each. Both models use Glorot uniform initialization [19]. For CIFAR-10, we utilize DenseNet121 and DenseNet201 [28], as implemented in Keras [10], with the addition of dropout regularization layers at rate 0.2 and weight decay of 10^{-4}, as prescribed in [28]. The DenseNet121 and DenseNet201 models have 6.9 million and 18 million parameters, respectively, and are both initialized with He normal [24].

Lastly, we explore a transfer learning scenario on the CIFAR-100 dataset [36], a choice reflecting the DL community's growing preference of using pre-trained models in such downstream tasks [23, 53]. For example, a pre-trained visual transformer (ViT) on ImageNet, transferred to classify CIFAR-100, is currently on par with the state-of-the-art results for this task [17]. We adopt this exact transfer learning scenario, leveraging the more powerful ConvNeXtLarge model, pre-trained on ImageNet, with 198 million parameters [10, 42]. Following the feature extraction step [20, 80], the testing accuracy on CIFAR-100 stands at 60%. Subsequently, we employ and evaluate our FDA algorithms in the arduous fine-tuning stage, where the entirety of the model is trained [75].

Algorithms. We consider five distributed deep learning algorithms: LinearFDA, SketchFDA, Synchronous [2], FedAdam [52], and FedAvgM [27]; the first three are standard in all experiments. Depending on the local optimizer, Adam [33] or SGD with Nesterov momentum (SGD-NM) [66], we also include their federated counterparts FedAdam or FedAvgM, respectively.

Evaluation Methodology. Comparing DDL algorithms is not straightforward. For example, comparing DDL algorithms based on the average cost of a training epoch can be misleading, as it does not consider the effects on the trained model's quality. To achieve a comprehensive performance assessment of FDA, we define a *training run* as the process of executing the DDL algorithm under evaluation, on (a) a specific DL model and training dataset, and (b) until a final epoch in which the trained model achieves a specific *testing accuracy* (termed as *Accuracy Target* in the Figures). Based on this definition, we focus on two performance metrics:

(1) **Communication cost**, which is the total data (in bytes) transmitted by all workers. Translating this cost to *wall-clock time* (i.e., the total time required for the computation and communication of the DDL) depends on the network infrastructure connecting the workers. Its impact is larger in FL scenarios, where workers often use slower Wi-Fi connections.

(2) **Computation cost**, which is the number of mini-batch steps (termed as *In-Parallel Learning Steps* in the Figures) performed by each worker. Translating this cost to *wall-clock time* is determined by the mini-batch size and the computational resources of the nodes. Its impact is larger for nodes with lower computational resources.

---

[2]The name was derived from the Bulk Synchronous Parallel approach; can be understood as a special case of the FDA Algorithm 1 where Θ is set to zero.

Table 1: Summary of Experiments

NN	d	Dataset	Hyper-Parameters			Training	
			Θ	b	K	Optimizer	Algorithms
LeNet-5	62K	MNIST	{0.5, 1, 1.5, 2, 3, 5, 7}	32	{ 5, 10, …, 60 }	Adam	FDA, SYNCHRONOUS, FEDADAM
VGG16*	2.6M	MNIST	{20, 25, 30, 50, 75, 90, 100}	32	{ 5, 10, …, 60 }	Adam	FDA, SYNCHRONOUS, FEDADAM
DenseNet121	6.9M	CIFAR-10	{200, 250, 275, 300, 325, 350, 400}	32	{ 5, 10, …, 30 }	SGD-NM	FDA, SYNCHRONOUS, FEDAVGM
DenseNet201	18M	CIFAR-10	{350, 500, 600, 700, 800, 850, 900}	32	{ 5, 10, …, 30 }	SGD-NM	FDA, SYNCHRONOUS, FEDAVGM
(fine-tuning) ConvNeXtLarge	198M	CIFAR-100	{25, 50, 100, 150}	32	{ 3, 5 }	AdamW	FDA, SYNCHRONOUS

Hyper-Parameters & Optimizers. Hyper-parameters unique to each training dataset and model are detailed in Table 1; Θ is pertinent to FDA algorithms and not applicable to others. Notably, a guideline for setting the parameter Θ is provided in Section 4.2. For experiments involving FEDAVGM and FEDADAM, we use $E = 1$ local epochs, following [52]. For experiments with LeNet-5 and VGG16*, local optimization employs Adam, using the default settings as per [33]. In these cases, FEDADAM also adheres to the default settings for both local and server optimization [10, 52]. For DenseNet121 and DenseNet201, local optimization is performed using SGD with Nesterov momentum (SGD-NM), setting the momentum parameter at 0.9 and learning rate at 0.1 [28]. For FEDAVGM, local optimization is conducted with default settings [10, 27], while server optimization employs SGD with momentum, setting the momentum parameter and learning rate to 0.9 and 0.316, respectively [52]. Lastly, for the transfer learning experiments, local optimization leverages AdamW [43], with the hyper-parameters used for fine-tuning ConvNeXtLarge in the original study [42].

Data Distribution. In all experiments, the training dataset is divided into approximately equal parts among the workers. To assess the impact of data heterogeneity, we explore three scenarios:

(1) **IID** — Independent and identically distributed.
(2) **Non-IID:** $X\%$ — A portion $X\%$ of the dataset is sorted by label and sequentially allocated to workers, with the remainder distributed in an IID fashion.
(3) **Non-IID: Label** Y — All samples from label Y are assigned to a few workers, while the rest are distributed in an IID manner.

Main Findings. The main findings of our experimental analyses are: (1) LINEARFDA and SKETCHFDA outperform the SYNCHRONOUS, FEDADAM and FEDAVGM techniques (their use depends on the local optimizer choice) by 1-2 orders of magnitude in communication, while maintaining equivalent model performance. (2) LINEARFDA and SKETCHFDA also significantly outperform the FEDADAM and FEDAVGM techniques in terms of computation. (3) The performance of LINEARFDA and SKETCHFDA is comparable in most experiments. SKETCHFDA provides a more accurate estimator of the variance and leads to fewer synchronizations than LINEARFDA, but has a larger communication overhead for its local state (a sketch, compared to a number). SKETCHFDA significantly outperforms LINEARFDA at the transfer learning scenario. (4) The FDA variants remain robust at various data heterogeneity settings, maintaining comparable performance to the IID case.

4.2 Results

Due to the extensive set of unique experiments (over 1000), as detailed in Table 1, we leverage Kernel Density Estimation (KDE) plots [74] to visualize the bivariate distribution of computation and communication costs incurred by each strategy for attaining the *Accuracy Target*. These KDE plots provide a high-level overview of the cost trade-off for training accurate models. As an illustrative example, Figure 1 depicts the strategies' bivariate distribution for the LeNet-5 model trained on MNIST with different data heterogeneity setups. In these plots, the SKETCHFDA distribution is generated from experiments across all hyper-parameter combinations (Θ and K in Table 1) that attained the *Accuracy Target* of 0.985.

FDA balances Communication vs. Computation. DDL algorithms face a fundamental challenge: balancing the competing demands of computation and communication. Frequent communication accelerates convergence and potentially improves model performance, but incurs higher network overhead, an overhead that may be prohibitive when workers communicate through lower speed connections. Conversely, reducing communication saves bandwidth but risks hindering, or even stalling, convergence. Traditional DDL approaches, like SYNCHRONOUS, require synchronizing model parameters after every learning step, leading to significant communication overhead but facilitating faster convergence (lower computation cost). This is evident in Figures 1, 2, 3, and 4 (where SYNCHRONOUS appears in the bottom right — low computation, very high communication). Conversely, Federated Optimization (FEDOPT) methods [52] are designed to be communication-efficient, reducing communication between devices (workers) at the expense of increased local computation. Indeed, as shown in Figures 1-4, FEDAVGM and FEDADAM reduce communication by orders of magnitude but at the price of a corresponding increase in computation. Our two proposed FDA strategies achieve the best of both worlds: the low computation cost of traditional methods and the communication efficiency of FEDOPT approaches, as seen in Figures 1-4. In fact, they significantly outperform FEDAVGM and FEDADAM in their element, that is, communication-efficiency. Across all experiments, the FDA methods' distributions lie in the desired bottom left quadrant — low computation, very low communication.

FDA counters diminishing returns. The phenomenon of *diminishing returns* states that as a DL model nears its learning limits for a given dataset and architecture, each additional increment in accuracy may necessitate a disproportionate increase in training time, tuning, and resources [20, 68]. We first clearly notice this with VGG16* on MNIST in Figure 2 for all three data heterogeneity
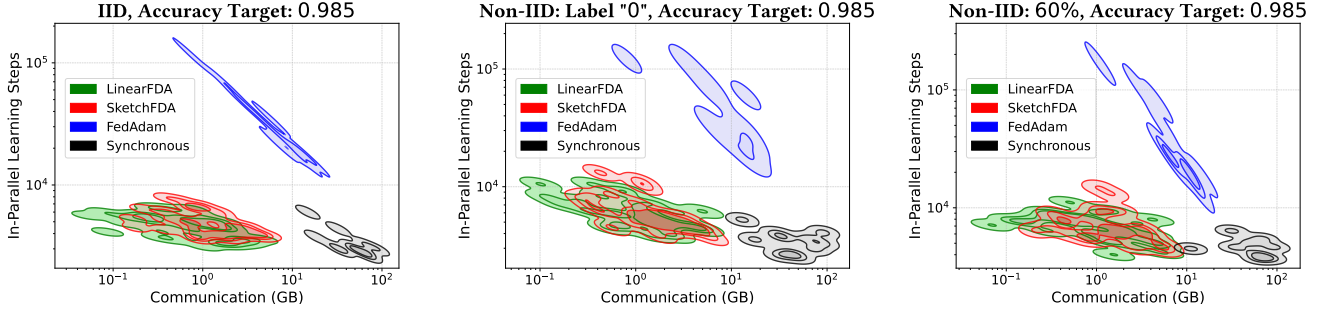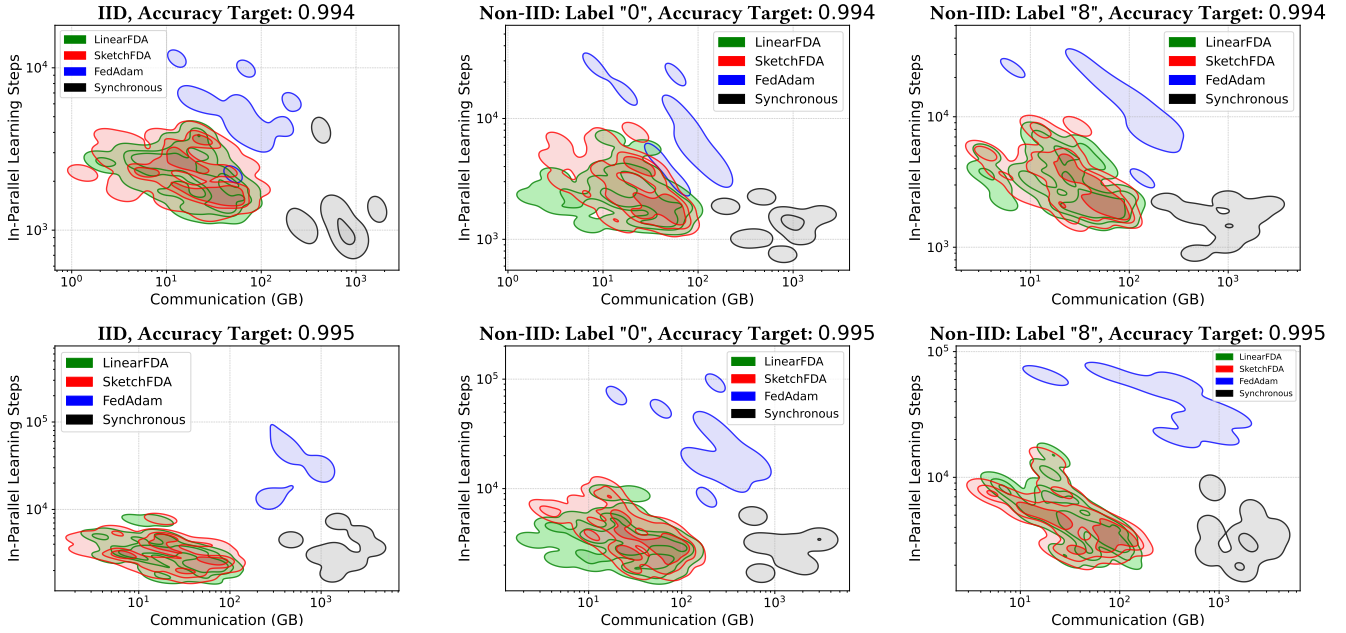
Figure 1: LeNet-5 on MNIST

Figure 2: VGG16* on MNIST

settings. For a 0.001 increase in accuracy (effectively 10 misclassified testing images) FEDADAM needs an order of magnitude more computation and communication. Similarly, SYNCHRONOUS requires comparable increases in computation and approximately half an order of magnitude more in communication. On the other hand, the FDA methods suffer a slight (if any) increase in computation and communication for this accuracy enhancement. For DenseNet121 and DenseNet201 on CIFAR-10 (Figures 3, 4), FEDAVGM and SYNCHRONOUS require half an order of magnitude more computation and communication to achieve the final marginal accuracy gains (0.78 to 0.81 for DenseNet121, and 0.78 to 0.8 for DenseNet201). In contrast, the FDA methods have almost no increase in communication and comparable increase in computation.

FDA is resilient to data heterogeneity. In DDL, data heterogeneity is a prevalent challenge, reflecting the complexity of real-world applications where the IID assumption often does not hold. The ability of DDL algorithms to maintain consistent performance in the face of non-IID data is a critical metric for their effectiveness and adaptability. Our empirical investigation reveals the FDA methods' noteworthy resilience in such heterogeneous environments. For LeNet-5 on MNIST, as illustrated in Figure 1, the computation and communication costs required to attain a test accuracy of 0.985 show negligible differences across the IID and the two Non-IID settings (Label "0", 60%). Similarly, for VGG16* on MNIST, Figure 2 demonstrates that achieving a test accuracy of 0.995 incurs comparable computation and communication costs across the IID and the two Non-IID settings (Label "0", Label "8"); while overall costs are aligned, the distributions of the computation costs exhibit greater variability, yet remain closely consistent with the IID scenario.

FDA generalizes better. The factors determining how well a DL algorithm performs are its ability to: (1) make the training accuracy high, and (2) make the gap between training and test accuracy small. These two factors correspond to the two central challenges in
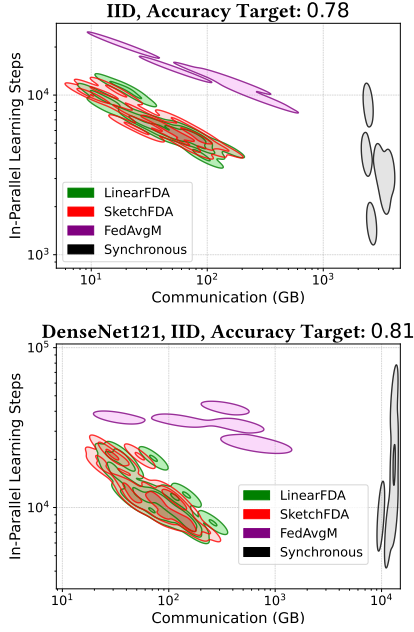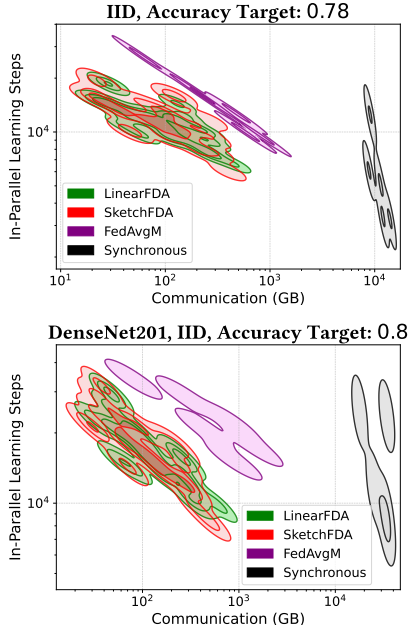
Figure 3: DenseNet121 on CIFAR-10

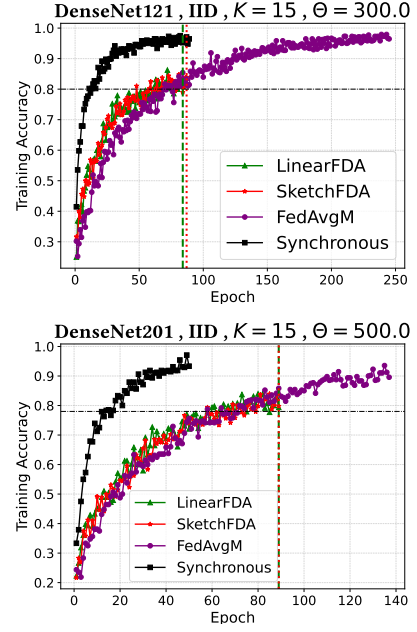Figure 4: DenseNet201 on CIFAR-10

Figure 5: Training accuracy progression with a test accuracy target (horizontal line) of 0.8 (left), and 0.78 (right). Dashed and doted lines indicate when LinearFDA and SketchFDA attain the target accuracy, respectively. A smaller final gap between training and target accuracy indicates less overfitting, i.e., better generalization capabilities of the trained model

DL: underfitting and overfitting [20]. For DenseNet121 on CIFAR-10, with a test accuracy target of 0.8, as illustrated in Figure 5, Synchronous and FedAvgM exhibit overfitting, with a noticeable discrepancy between training and test accuracy. In stark contrast, the FDA methods have an almost zero accuracy gap. Please note that LinearFDA and SketchFDA reach the test accuracy target

of 0.8 much earlier (at epochs 86 and 91, respectively). Turning our focus to DenseNet201 on CIFAR-10, with a test accuracy target of 0.78, Synchronous again tends towards overfitting, while FedAvgM shows a slight improvement but still does not match the FDA methods, which continue to exhibit exceptional generalization capabilities, evidenced by a minimal training-test accuracy gap, as shown at Figure 5. Notably, given the necessity to fix hyperparameters Θ and K for the training accuracy plots, we selected two representative examples. The patterns of performance we highlighted are consistent across most of the conducted tests.

Dependence on K. In distributed computing, scaling up typically results in proportional speed improvements. In DDL, however, scalability is less predictable due to the nuanced interplay of computation and communication costs with convergence, complicating the expected linear speedup. This unpredictability is starkly illustrated with VGG16* on the MNIST dataset across all data heterogeneity settings and all strategies. Specifically, Figure 6 demonstrates (at its upper part) that increasing the number of workers does not decrease computation – except for FedAdam which begins with significantly high computation – but rather exacerbates communication (shown at the upper part of the figure). These findings are troubling, as they reveal scaling up only hampers training speed and wastes resources. However, for more complex learning tasks like training DenseNet-121 and DenseNet-201 on CIFAR-10 (upper part of Figures 7, 8), the expected behavior starts to emerge. Specifically, for DenseNet-121, scaling up (K increase) leads to a
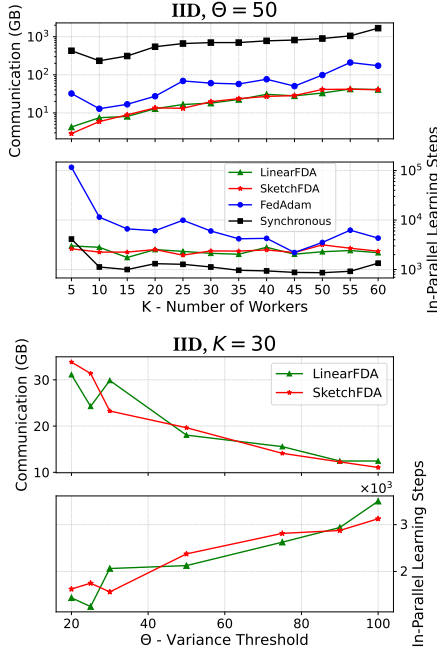
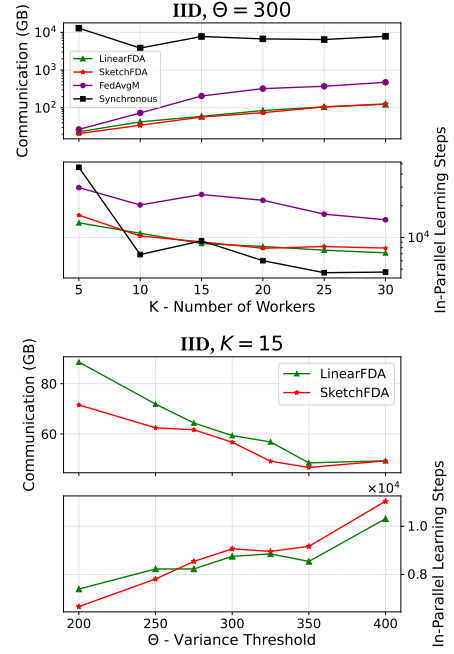Figure 6: VGG16* on MNIST — Accuracy Target: 0.994

Figure 7: DenseNet121 on CIFAR-10 — Accuracy Target: 0.8

decrease in computation cost for all strategies. Communication cost, however, increases with K for all methods except Synchronous, which maintains constant communication irrespective of worker count, but at the expense of orders of magnitude higher communication overhead. Notably, while our findings might, in some cases, suggest potential speed benefits of not scaling up (smaller K), DDL is increasingly conducted within federated settings, where there is no other choice but to utilize the high number of workers.

FDA: Dependence on Θ. The variance threshold Θ can be seen as a lever in balancing communication and computation; essentially, it calibrates the trade-off between these two costs. A higher Θ allows for greater model divergence before synchronization, reducing communication at the cost of potentially increased computation to achieve convergence. This impact of Θ is consistently observed across all two FDA strategies, learning tasks, and data heterogeneity settings (Figures 6, 7 and 8). Interestingly, for more complex models like DenseNet121 and DenseNet201 on CIFAR-10, increasing the variance threshold (Θ) does not lead to a significant rise in computation cost, as illustrated in Figures 7 and 8. It suggests that the FDA methods, by strategically timing synchronizations (monitoring the variance), substantially reduce the number of necessary synchronizations without a proportional increase in computation for the same model performance; this is particularly promising for complex DDL tasks.

FDA: Choice of Θ. The experimental results suggest that selecting any Θ within a specific order of magnitude (e.g., between 10^2 and 10^3 for DenseNet201) ensures convergence, as demonstrated in Figures 6, 7, and 8. Therefore, identifying this range becomes crucial. To this end, we conducted extensive exploratory testing

to estimate the Θ ranges for each learning task which are predominantly influenced by the number of parameters d of the DNN. Within this context, Θ values outside the desirable range exhibit notable effects: below this range, the training process mimics Synchronous or Local-SGD approaches with small τ, while exceeding it leads to non-convergence. Subsequently, having identified the optimal ranges for Θ, we selected diverse values within them for our experimental evaluation (Table 1), thereby investigating different computation and communication trade-offs. For instance, in the ARIS-HPC environment with an InfiniBand connection (up to 56 Gb/s), experiments show that training wall-time (the total time required for the computation and the communication of the DDL) is predominantly influenced by computation cost, rendering communication concerns negligible. In such contexts, lower Θ values are favored due to their computational efficiency. On the contrary, in FL settings, where communication typically poses the greater challenge, opting for higher Θ values proves advantageous; reduction in communication achieved with higher Θ values will translate in a large reduction in total wall-time.

To assist researchers in selecting an optimal variance threshold, Figure 9 presents empirical estimations for Θ across three distinct learning settings: FL (assuming a common channel of 0.5Gbps), Balanced (communication-computation equilibrium), and HPC.

$$\Theta_{FL} = 4.91 \cdot 10^{-5} \cdot d$$
$$\Theta_{B} = 3.89 \cdot 10^{-5} \cdot d$$
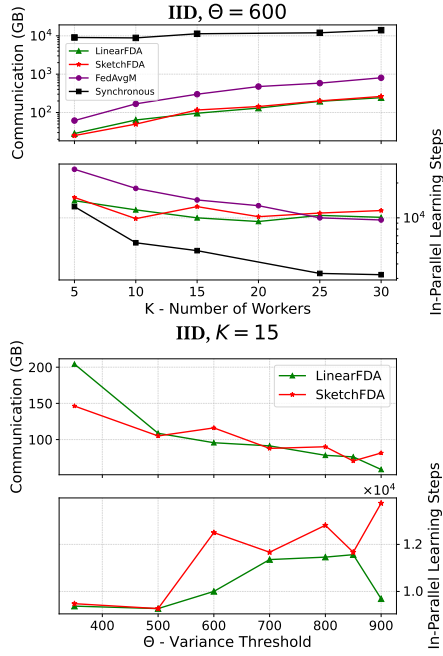$$\Theta_{HPC} = 2.74 \cdot 10^{-5} \cdot d$$

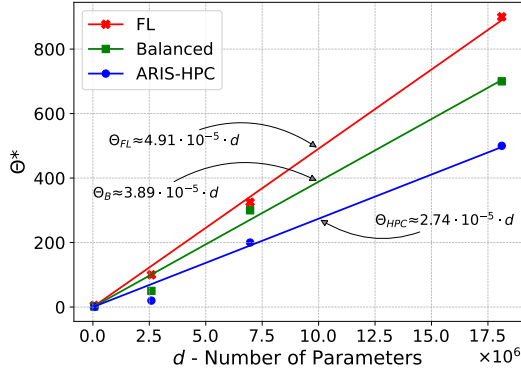Figure 8: DenseNet201 on CIFAR-10 — Accuracy Target: 0.78

Figure 9: Empirical Estimation of the Variance Threshold

Figure 10: ConvNeXtLarge on CIFAR-100 (transfer learning from ImageNet) — Deployment of FDA during the fine-tuning stage with Accuracy Target of 0.76

computation. On the other hand, for intricate learning tasks and deeper models, SKETCHFDA becomes the preferred choice, particularly if communication-efficiency is paramount.

5 CONCLUSIONS

In this paper, we introduced Federated Dynamic Averaging (FDA), an innovative, adaptive and communication-efficient algorithm for distributed deep learning. Essentially, FDA makes informed, dynamic decisions on when to terminate training rounds based on approximations of the model variance. Through extensive experiments across diverse datasets and learning tasks, we demonstrated that FDA significantly reduces communication overhead (often by orders of magnitude) without a corresponding increase in computation or compromise in model performance — contrary to the typical trade-offs encountered in the literature. Furthermore, we showed that FDA is robust to data heterogeneity and inherently mitigates overfitting. Our results push the limits of modern communication-efficient distributed deep learning, paving the way for scalable, dynamic, and broadly applicable strategies.

FDA: Linear vs. Sketch. In our main body of experiments, across most learning tasks and data heterogeneity settings, the two proposed FDA methods exhibit comparable performance, as illustrated in Figures 1, 2, 3, and 4. This suggests that the precision of the variance approximation is not critical; occasional "unnecessary" synchronizations do not significantly impact overall performance. However, in all experiments within the more intricate transfer learning scenario, LINEARFDA requires approximately 1.5 times more communication than SKETCHFDA to fine-tune the deep ConvNeXt-Large model to equivalent performance levels (Figure 10).

In light of these findings, we conclude the following: for straightforward and less demanding tasks, LINEARFDA is the recommended option due to its simplicity and lower complexity per local state
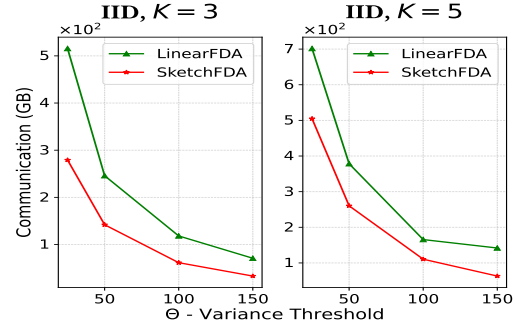
REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

[2] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. 2021. Federated Learning Based on Dynamic Regularization. *CoRR* abs/2111.04263 (2021). arXiv:2111.04263

[3] Alham Fikri Aji and Kenneth Heafield. 2017. Sparse Communication for Distributed Gradient Descent. *CoRR* abs/1704.05021 (2017). arXiv:1704.05021

[4] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Sarit Khirirat, Nikola Konstantinov, and Cédric Renggli. 2018. The Convergence of Sparsified Gradient Methods. *CoRR* abs/1809.10505 (2018). arXiv:1809.10505

[5] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. 2015. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *CoRR* abs/1512.02595 (2015). arXiv:1512.02595

[6] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. 2019. Qsparse-local-SGD: Distributed SGD with Quantization, Sparsification, and Local Computations. arXiv:1906.02367 [stat.ML]

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *CoRR* abs/2005.14165 (2020). arXiv:2005.14165

[8] Tianyi Chen, Georgios B. Giannakis, Tao Sun, and Wotao Yin. 2018. LAG: Lazily Aggregated Gradient for Communication-Efficient Distributed Learning. arXiv:1805.09965 [stat.ML]

[9] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. 2014. Project Adam: Building an Efficient and Scalable Deep Learning Training System. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, Broomfield, CO, 571–582.

[10] François Chollet et al. 2015. Keras.

[11] Graham Cormode and Minos Garofalakis. 2005. Sketching Streams through the Net: Distributed Approximate Query Tracking. In *Proceedings of the 31st International Conference on Very Large Data Bases* (Trondheim, Norway) *(VLDB '05)*. VLDB Endowment, 13–24.

[12] Henggang Cui, James Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Abhimanu Kumar, Jinliang Wei, Wei Dai, Gregory R. Ganger, Phillip B. Gibbons, Garth A. Gibson, and Eric P. Xing. 2014. Exploiting Bounded Staleness to Speed Up Big Data Analytics. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. USENIX Association, Philadelphia, PA, 37–48.

[13] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. 2012. Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc.

[14] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. 2010. Optimal Distributed Online Prediction using Mini-Batches. *CoRR* abs/1012.1367 (2010). arXiv:1012.1367

[15] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR* abs/2010.11929 (2020). arXiv:2010.11929

[18] Sanghamitra Dutta, Gauri Joshi, Soumyadip Ghosh, Parijat Dube, and Priya Nagpurkar. 2018. Slow and Stale Gradients Can Win the Race: Error-Runtime Trade-offs in Distributed SGD. arXiv:1803.01113 [stat.ML]

[19] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Yee Whye Teh and Mike Titterington (Eds.), Vol. 9. PMLR, Chia Laguna Resort, Sardinia, Italy, 249–256.

[20] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA, USA.

[21] Suyog Gupta, Wei Zhang, and Fei Wang. 2016. Model Accuracy and Runtime Tradeoff in Distributed Deep Learning:A Systematic Study. arXiv:1509.04210 [stat.ML]

[22] Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck R. Cadambe. 2019. Local SGD with Periodic Averaging: Tighter Analysis and Adaptive Synchronization. *CoRR* abs/1910.13598 (2019). arXiv:1910.13598

[23] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-Trained Models: Past, Present and Future. *CoRR* abs/2106.07139 (2021). arXiv:2106.07139

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR* abs/1502.01852 (2015). arXiv:1502.01852

[25] Qirong Ho, James Cipar, Henggang Cui, Jin Kyu Kim, Seunghak Lee, Phillip B. Gibbons, Garth A. Gibson, Gregory R. Ganger, and Eric P. Xing. 2013. More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1* (Lake Tahoe, Nevada) *(NIPS'13)*. Curran Associates Inc., Red Hook, NY, USA, 1223–1231.

[26] Elad Hoffer, Itay Hubara, and Daniel Soudry. 2018. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv:1705.08741 [stat.ML]

[27] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. arXiv:1909.06335 [cs.LG]

[28] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016. Densely Connected Convolutional Networks. *CoRR* abs/1608.06993 (2016). arXiv:1608.06993

[29] Rie Johnson and Tong Zhang. 2013. Accelerating Stochastic Gradient Descent Using Predictive Variance Reduction. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1* (Lake Tahoe, Nevada) *(NIPS'13)*. Curran Associates Inc., Red Hook, NY, USA, 315–323.

[30] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2019. Advances and Open Problems in Federated Learning. *CoRR* abs/1912.04977 (2019). arXiv:1912.04977

[31] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. 2021. Mime: Mimicking Centralized Stochastic Algorithms in Federated Learning. arXiv:2008.03606 [cs.LG]

[32] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. 2019. SCAFFOLD: Stochastic Controlled Averaging for On-Device Federated Learning. *CoRR* abs/1910.06378 (2019). arXiv:1910.06378

[33] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[34] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. 2020. Big Transfer (BiT): General Visual Representation Learning. arXiv:1912.11370 [cs.CV]

[35] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *CoRR* abs/1610.02527 (2016). arXiv:1610.02527

[36] Alex Krizhevsky. 2012. Learning Multiple Layers of Features from Tiny Images. *University of Toronto* (05 2012).

[37] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[38] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. 2020. PyTorch Distributed: Experiences on Accelerating Data Parallel Training. *CoRR* abs/2006.15704 (2020). arXiv:2006.15704

[39] Xiang Li, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2021. Communication-Efficient Local Decentralized SGD Methods. arXiv:1910.09126 [stat.ML]

[40] Tao Lin, Sebastian U. Stich, and Martin Jaggi. 2018. Don't Use Large Mini-Batches, Use Local SGD. *CoRR* abs/1808.07217 (2018). arXiv:1808.07217

[41] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. 2017. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. *CoRR* abs/1712.01887 (2017). arXiv:1712.01887

[42] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A ConvNet for the 2020s. *CoRR* abs/2201.03545 (2022). arXiv:2201.03545

[43] Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. *CoRR* abs/1711.05101 (2017). arXiv:1711.05101

[44] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629 (2016). arXiv:1602.05629

[45] Jed Mills, Jia Hu, and Geyong Min. 2023. Faster Federated Learning with Decaying Number of Local SGD Steps. arXiv:2305.09628 [cs.LG]

[46] Ioannis Mitliagkas, Ce Zhang, Stefan Hadjis, and Christopher Ré. 2016. Asynchrony begets Momentum, with an Application to Deep Learning. arXiv:1605.09774 [stat.ML]

[47] Philipp Moritz, Robert Nishihara, Ion Stoica, and Michael I. Jordan. 2016. SparkNet: Training Deep Networks in Spark. arXiv:1511.06051 [stat.ML]

[48] Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. 2011. HOG-WILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. arXiv:1106.5730 [math.OC]

[49] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. 2012. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence* 2 (03 2012).

[50] Tiancheng Qin, S. Rasoul Etesami, and César A. Uribe. 2023. The role of local steps in local SGD. *Optimization Methods and Software* (Aug. 2023), 1–27.

[51] M.G. Rabbat and R.D. Nowak. 2005. Quantized incremental algorithms for distributed optimization. *IEEE Journal on Selected Areas in Communications* 23, 4 (2005), 798–808.

[52] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. 2021. Adaptive Federated Optimization. arXiv:2003.00295 [cs.LG]

[53] Tal Ridnik, Emanuel Ben Baruch, Asaf Noy, and Lihi Zelnik-Manor. 2021. ImageNet-21K Pretraining for the Masses. *CoRR* abs/2104.10972 (2021). arXiv:2104.10972

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575 [cs.CV]

[55] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. 2018. On the Convergence of Federated Optimization in Heterogeneous Networks. *CoRR* abs/1812.06127 (2018). arXiv:1812.06127

[56] Christopher J. Shallue, Jaehoon Lee, Joseph M. Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. 2018. Measuring the Effects of Data Parallelism on Neural Network Training. *CoRR* abs/1811.03600 (2018). arXiv:1811.03600

[57] Shaohuai Shi and Xiaowen Chu. 2017. Performance Modeling and Evaluation of Distributed Deep Learning Frameworks on GPUs. *CoRR* abs/1711.05979 (2017). arXiv:1711.05979

[58] Shaohuai Shi, Zhenheng Tang, Xiaowen Chu, Chengjian Liu, Wei Wang, and Bo Li. 2021. A Quantitative Survey of Communication Optimizations in Distributed Deep Learning. *IEEE Network* 35, 3 (2021), 230–237.

[59] Nir Shlezinger, Mingzhe Chen, Yonina C. Eldar, H. Vincent Poor, and Shuguang Cui. 2020. UVeQFed: Universal Vector Quantization for Federated Learning. *CoRR* abs/2006.03262 (2020). arXiv:2006.03262

[60] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, Article arXiv:1409.1556 (Sept. 2014), arXiv:1409.1556 pages. arXiv:1409.1556 [cs.CV]

[61] Virginia Smith, Simone Forte, Chenxin Ma, Martin Takáč, Michael I. Jordan, and Martin Jaggi. 2016. CoCoA: A General Framework for Communication-Efficient Distributed Optimization. *CoRR* abs/1611.02189 (2016). arXiv:1611.02189

[62] Sebastian U. Stich. 2019. Local SGD Converges Fast and Communicates Little. arXiv:1805.09767 [math.OC]

[63] Nikko Ström. 2015. Scalable Distributed DNN Training Using Commodity GPU Cloud Computing. In *Interspeech 2015*.

[64] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *CoRR* abs/1707.02968 (2017). arXiv:1707.02968

[65] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Sanjoy Dasgupta and David McAllester (Eds.), Vol. 28. PMLR, Atlanta, Georgia, USA, 1139–1147.

[66] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Sanjoy Dasgupta and David McAllester (Eds.), Vol. 28. PMLR, Atlanta, Georgia, USA, 1139–1147.

[67] Zhenheng Tang, Shaohuai Shi, Bo Li, and Xiaowen Chu. 2023. GossipFL: A Decentralized Federated Learning Framework With Sparsified and Adaptive Communication. *IEEE Transactions on Parallel and Distributed Systems* 34, 3 (2023), 909–922.

[68] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. 2021. Deep Learning's Diminishing Returns: The Cost of Improvement is Becoming Unsustainable. *IEEE Spectrum* 58, 10 (2021), 50–55.

[69] Leslie G. Valiant. 1990. A Bridging Model for Parallel Computation. *Commun. ACM* 33, 8 (aug 1990), 103–111.

[70] Jianyu Wang and Gauri Joshi. 2018. Adaptive Communication Strategies to Achieve the Best Error-Runtime Trade-off in Local-Update SGD. *CoRR* abs/1810.08313 (2018). arXiv:1810.08313

[71] Jianyu Wang and Gauri Joshi. 2021. Cooperative SGD: A Unified Framework for the Design and Analysis of Local-Update SGD Algorithms. *Journal of Machine Learning Research* 22, 213 (2021), 1–50.

[72] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221.

[73] Zeqin Wang, Ming Wen, Yuedong Xu, Yipeng Zhou, Jessie Hui Wang, and Liang Zhang. 2023. Communication compression techniques in distributed deep learning: A survey. *Journal of Systems Architecture* 142 (2023), 102927.

[74] Michael L. Waskom. 2021. seaborn: statistical data visualization. *Journal of Open Source Software* 6, 60 (2021), 3021.

[75] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *CoRR* abs/1411.1792 (2014). arXiv:1411.1792

[76] Hao Yu and Rong Jin. 2019. On the Computation and Communication Complexity of Parallel SGD with Dynamic Batch Sizes for Stochastic Non-Convex Optimization. arXiv:1905.04346 [math.OC]

[77] Hao Yu, Sen Yang, and Shenghuo Zhu. 2018. Parallel Restarted SGD with Faster Convergence and Less Communication: Demystifying Why Model Averaging Works for Deep Learning. arXiv:1807.06629 [math.OC]

[78] Jian Zhang, Christopher De Sa, Ioannis Mitliagkas, and Christopher Ré. 2016. Parallel SGD: When does averaging help? arXiv:1606.07365 [stat.ML]

[79] Fan Zhou and Guojing Cong. 2017. On the convergence properties of a K-step averaging stochastic gradient descent algorithm for nonconvex optimization. *CoRR* abs/1708.01012 (2017). arXiv:1708.01012

[80] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2019. A Comprehensive Survey on Transfer Learning. *CoRR* abs/1911.02685 (2019). arXiv:1911.02685

[81] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex Smola. 2010. Parallelized Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Eds.), Vol. 23. Curran Associates, Inc.