Communication-Efficient Distributed Deep Learning via Federated Dynamic Averaging

Michail Theologitis
Technical University of Crete
Chania, Greece
mtheologitis@tuc.gr

Georgios Frangias
Technical University of Crete
Chania, Greece
gfrangias@tuc.gr

Georgios Anestis
Technical University of Crete
Chania, Greece
ganestis@tuc.gr

Vasilis Samoladas
Technical University of Crete
Chania, Greece
vsamoladas@tuc.gr

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

ABSTRACT

Driven by the ever-growing volume and decentralized nature of data, coupled with the need to harness this data and generate knowledge from it, has led to the extensive use of distributed deep learning (DDL) techniques for training. These techniques rely on local training that is performed at the distributed nodes based on locally collected data, followed by a periodic synchronization process that combines these models to create a global model. 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. These drawbacks also have a direct impact on the time required for the training process, necessitating excessive time for data communication. 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. In essence, the costly synchronization step is triggered only if the local models, which are initialized from a common global model after each synchronization, have significantly diverged. This decision is facilitated by the communication of a small local state from each distributed node/worker. 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. Additionally, we show that FDA maintains robust performance across diverse data heterogeneity settings.

1 INTRODUCTION

The big data era has been marked by an unprecedented scale of training datasets [41, 67]. 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 (especially in federated settings, such as drones collecting and collaboratively building a global model/view of an area), and data sovereignty laws [9, 23, 64]. Such constraints complicate the use of Deep Learning (DL) techniques in the aforementioned scenarios.

Distributed Deep Learning (DDL) has emerged as an alternative paradigm to the traditional centralized approach [6, 69], offering efficient learning over 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 [10, 28, 35, 55, 68]. Most DDL methods are iterative, where, in each iteration, some amount of local training is followed by *synchronization* of the local models with the global one. The predominant method, based on the bulk synchronous parallel (BSP) approach [56], is to average the local model updates and then apply the average update to each local model [69]. Less synchronized variants have also been proposed, to ameliorate the effect of *straggler workers* [14, 37] but compromise convergence speed and model quality.

A significant 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 [53, 60]. Specifically, the communication bottleneck arises from the frequent exchange (synchronization) of model parameters, often in the range of billions, across distributed workers. The synchronization process entails substantial data volume transfer and generally dominates the overall training time, leading to a low computation-to-communication ratio [14, 46]. Addressing this challenge 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 [58].

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 [17, 66]. Although Local-SGD is effective in reducing communication while maintaining comparable model quality [58], determining the optimal value of τ presents a critical challenge, with only a handful of studies offering theoretical insights into its influence on convergence [50, 58, 66].

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 [57], 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 [17] was on reducing the number of communication rounds for a fixed number of model updates and an increasing sequence emerged. 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 Idea and Contributions. Our work addresses critical efficiency challenges in DDL, particularly in communication-constrained environments, such as the ones encountered in Federated Learning (FL) applications [23]. We introduce Federated Dynamic Averaging (FDA), a novel, adaptive distributed deep learning strategy that massively improves communication efficiency over previous work.

FDA utilizes a novel 2-action, conditional synchronization protocol, designed to avoid the need to decide or guess the proper values of local update steps, or to synchronize after each training step, but rather only performs the costly synchronization process *when needed*. Our FDA algorithm dynamically triggers synchronization based on the value of *model variance* across worker-nodes. In a nutshell, the costly synchronization step is only triggered if the local models have diverged significantly, which implies that the global model may no longer be accurate.

As Figure 1 demonstrates, at the start, workers enter the local training step with the same global model (Figure 1.A). Then, local training commences and each distributed worker-node computes its local state, which encapsulates helpful information for estimating the model variance (Figure 1.B). This is followed by the transmission (Figure 1.C) of these small-size local states, an operation that is bandwidth- and time-efficient because of their small size. During transmission, the local states are aggregated and their average is made available to all workers—an operation known as AllReduce. This operation does not require (or prohibit) the use of a central node. Based on the aggregated state, the workers can estimate (Figure 1.D) whether the variance of the local models may have exceeded a threshold. If this is not the case, the costly synchronization step (Figure 1.E) is avoided and local training continues. What is important is how to properly pick these local states computed at, and then transmitted by, the local workers. To address this problem, we propose two variants of our FDA algorithm. Our contributions can be summarized as follows:

- We propose FDA, an algorithm that dynamically decides to synchronize local workers when *model variance across workers* exceeds a threshold. This strategy drastically reduces communication, while preserving cohesive progress towards the shared training objective.
- We propose two variants of FDA, which differ in the amount of information preserved in the local states that are transmitted by each worker and aggregated for subsequent estimation of model variance. These two variants, termed SketchFDA and LinearFDA, offer a different balance between communication efficiency and approximation accuracy.
- We evaluate and compare FDA with other DDL algorithms through a comprehensive suite of experiments with diverse datasets, models, and tasks. Our experiments demonstrate that FDA outperforms traditional and contemporary FL algorithms by 1-2 orders of magnitude in communication savings, while maintaining equivalent model performance. Furthermore, it effectively balances the competing demands of communication and computation, providing greatly improved trade-offs.
- We demonstrate FDA's robustness in various challenging Non-IID settings, common in real-world Federated Learning applications. While state-of-the-art methods typically require substantially more resources to converge under Non-IID conditions, FDA maintains consistent and comparable performance across both IID and Non-IID settings.

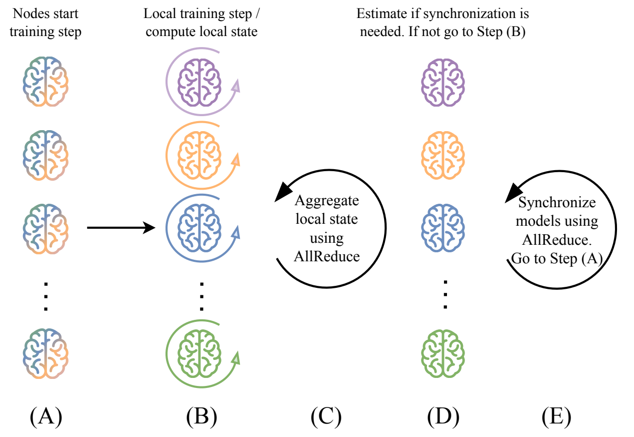

Nodes start training step
Local training step / compute local state
Estimate if synchronization is needed. If not go to Step (B)
Aggregate local state using AllReduce
Synchronize models using AllReduce. Go to Step (A)
(A)
(B)
(C)
(D)
(E)

Figure 1: FDA. The local training step is followed by the computation of a local state by all worker-nodes. Then, the (small in size) local states are aggregated. Based on the aggregated result, all workers estimate if synchronization is required. In most cases, the expensive synchronization step of the models is avoided and local training continues

Outline. The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 introduces our DDL technique, Federated Dynamic Averaging (FDA), and its two variants. Section 4 details the experimental setup, and discusses the insights and conclusions drawn from our empirical investigation. Lastly, Section 5 contains concluding remarks.

2 RELATED WORK

Problem formulation. Consider distributed training of deep neural networks over multiple workers [11, 31]. In this setting, each worker represents a data owner (equivalently, a local model owner) and has access to its own set of training data $\mathcal{D}_k$. Workers can utilize any available hardware they possess (e.g., GPUs, CPUs) to perform learning steps. 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:

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

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

Solution direction. As noted in the seminal work [23], research in FL should focus primarily on synchronous solutions. This allows different lines of research (e.g., compression, privacy, etc.) to be developed independently and then combined seamlessly. Our work, along with most communication-efficient FL strategies, adheres to this synchronous paradigm. However, such approaches may be less effective in environments where each communication operation incurs significant overhead regardless of the size of the data being transmitted (e.g., high-latency). In these scenarios, asynchronous mechanisms become necessary, though they typically fall outside the primary focus of contemporary FL research. That said, FDA can be modified to work asynchronously (as explained in Section 3.3).

Communication efficient Local-SGD. The work in [31] decomposes each round into two phases. In the first phase, each worker runs Local-SGD with $\tau = I_1$, while the second phase runs I_2 steps with $\tau = 1$; [31] proposes to exponentially decay I_1 every M rounds. In the heterogeneous setting, the work in [40], by analysing the convergence rate, proposes an increasing sequence of local update steps for strongly-convex local objectives and fixed local update steps for other types of local objectives. The study in [65] 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 [20], a challenge addressed by the post-local SGD method [32], which divides training into two phases: BSP-SGD followed by Local-SGD with a fixed number of steps. In the Lazily Aggregated Algorithm (LAG) [5], 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) [36] is another representative of communication efficient Local-SGD algorithms, which is a pivotal method in Federated Learning (FL) [23]. In the FL setting with edge computing systems, the work in [59] tries to find the optimal synchronization period τ subject to local computation and aggregation constraints. Recently [38], 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.

Unlike previous approaches that rely on predetermined synchronization schedules (fixed, decaying, or otherwise), our work introduces a dynamic synchronization strategy. FDA adapts continuously during the training process, basing synchronization decisions on a real-time metric: the model variance across workers.

Accelerating convergence. An indirect, yet highly effective way to mitigate the communication burden in DDL, is to speed up convergence. Consequently, recent works have built upon communication efficient Local-SGD methods by deploying accelerated versions of SGD to the distributed setting. Specifically, FedAdam [42] extends Adam [26] and FedAvgM [21] extends SGD with momentum (SGD-M) [51]. Recently, Mime [24] 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 [25, 42, 63]. The result is slow and unstable convergence with significant communication overhead. To address this problem, the SCAFFOLD algorithm [25] used control-variates (in the same spirit to SVRG), with significant speed-up. FedProx [45] re-parameterized FedAvg [36] 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.

While these approaches primarily focus on enhancing the optimization process and typically employ fixed synchronization intervals (e.g., every local epoch), our work addresses a complementary aspect: determining the optimal timing for synchronization. FDA's dynamic synchronization strategy is orthogonal to these optimization techniques and can be integrated with them by simply adjusting the synchronization decision.

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 [3], and quantization techniques, which involve transmitting only quantized gradients, as detailed in [47]. These techniques can be combined with Local-SGD methods to enhance communication-efficiency further. An example is Qsparse-local-SGD [4], which integrates aggressive sparsification and quantization with Local-SGD, achieving substantial communication savings. Crucially, FDA is fully compatible with any technique that reduces the cost of synchronization (e.g. model compression). Our approach simply adjusts the timing of the synchronization decision without altering the data being synchronized. This ensures that any compression technique effective in traditional methods (BSP, Local-SGD, etc.) will be equally effective when deployed with FDA. Therefore, the communication savings demonstrated in the relevant literature [61] can be safely expected to carry over to our approach as well.

Additionally, sketching emerges as another fundamental tool in large-scale machine learning. It effectively compresses high-dimensional problems into lower dimensions to save runtime and memory, typically utilizing hash-based probabilistic data structures. For instance, [49] use Count Sketches to compress auxiliary variables in optimization algorithms, significantly freeing up memory. Similarly, FetchSGD [43] employs Count Sketches to compress model updates and leverages their linearity for efficient merging. In contrast to these applications, our approach utilizes sketches not for compression but to estimate local state information, and based on this to decide whether a synchronization is required—an orthogonal application to traditional use cases. A comprehensive survey of compression techniques in DDL can be found in [61].

3 FEDERATED DYNAMIC AVERAGING

We now present our algorithms, based on our notion of Federated Dynamic Averaging (FDA). Our algorithms deviate from prior work in these two key ways:

(1) The decision on when to synchronize.
(2) The actual synchronization process.

To the best of our knowledge, this is the first Distributed Deep Learning algorithm that dynamically decides when to synchronize based on the current collective state of the training progress—whether it is advancing well or poorly.

Notation. At each time step t, each worker k independently maintains its own vector of model parameters[1], denoted as $\mathbf{w}_t^{(k)} \in \mathbb{R}^d$. Let $\mathbf{w}_t$ represent the $K \times d$ tensor of all local model vectors, and $\overline{\mathbf{w}}_t$ be the average model vector (this notation applies to all vector quantities):

$$\mathbf{w}_t = \left[\mathbf{w}_t^{(1)}, \ldots, \mathbf{w}_t^{(K)}\right] \quad , \quad \overline{\mathbf{w}}_t = \frac{1}{K}\sum_{k=1}^{K} \mathbf{w}_t^{(k)}$$

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

Table 1: Notation

Symbol	Meaning
$\langle \cdot, \cdot \rangle$	Dot product
t	Time step index
K	Number of workers
d	Model dimension
$\mathcal{D}_k$	Training data of worker k
$\mathcal{B}_t^{(k)}$	A batch sampled from $\mathcal{D}_k$
$\mathbf{w}_t^{(k)} \in \mathbb{R}^d$	Model of worker k
$\mathbf{w}_t = [\mathbf{w}_t^{(1)}, \dots, \mathbf{w}_t^{(K)}]$	Tensor of local models
$\overline{\mathbf{w}}_t = \frac{1}{K}\sum_{k=1}^K \mathbf{w}_t^{(k)}$	Average model (global model)
$\overline{\mathbf{w}}_{t_0}$	Model after most recent sync.
$\overline{\mathbf{w}}_{t_{-1}}$	Model after 2nd most recent sync.
$\mathbf{u}_t^{(k)} = \mathbf{w}_t^{(k)} - \overline{\mathbf{w}}_{t_0}$	Local model drift
$\overline{\mathbf{u}}_t = \frac{1}{K}\sum_{k=1}^K \mathbf{u}_t^{(k)}$	Average model drift (global drift)
$\mathcal{V}ar\left(\mathbf{w}_t\right)$	Model variance
Θ	Model variance threshold
$\mathrm{S}_t^{(k)}$	State of worker k
$\overline{\mathrm{S}}_t = \frac{1}{K}\sum_{k=1}^K \mathrm{S}_t^{(k)}$	Average state
$H(\cdot)$	Function for variance estimation
$\mathrm{sk}(\cdot) : \mathbb{R}^d \to \mathbb{R}^{l \times m}$	AMS sketch operator (§3.1)
$\mathcal{M}_2(\cdot) : \mathbb{R}^{l \times m} \to \mathbb{R}$	L^2 norm squared estimate (§3.1)
ϵ	Error of sketch estimate (§3.1)
$(1-\delta)$	Confidence of approximation (§3.1)
$l = O(\log 1/\delta)$	#Rows of sketch matrix (§3.1)
$m = O(1/\epsilon^2)$	#Columns of sketch matrix (§3.1)
$\xi = \frac{\overline{\mathbf{w}}_{t_0} - \overline{\mathbf{w}}_{t_{-1}}}{\lVert \overline{\mathbf{w}}_{t_0} - \overline{\mathbf{w}}_{t_{-1}} \rVert_2}$	Heuristic vec. for LinearFDA (§3.2)

Furthermore, let Optimize($\mathbf{w}, \mathcal{B}$) be the updated model [16] computed by some optimization algorithm (e.g., SGD, Adam) using the model $\mathbf{w}$, and the batch $\mathcal{B}$ of training data. It incorporates the learning rate, loss function and relevant gradients. During step t, each worker k first applies the update:

$$\mathbf{w}_t^{(k)} = \text{Optimize}(\mathbf{w}_{t-1}^{(k)}, \mathcal{B}_t^{(k)})$$

Moreover, operation AllReduce($\mathbf{w}_t^{(k)}$) computes and returns the average model vector [30]:

$$\overline{\mathbf{w}}_t = \text{AllReduce}(\mathbf{w}_t^{(k)})$$

Workers *synchronize* by executing AllReduce($\mathbf{w}_t^{(k)}$), thereby setting $\mathbf{w}_t^{(k)} := \overline{\mathbf{w}}_t$. If synchronization is not performed at step t, each worker continues training with its locally updated model. A comprehensive list of the notation used throughout this section is provided in Table 1.

Model Variance and FDA. The *model variance* quantifies the dispersion or spread of worker models around the average model:

$$\mathcal{V}ar\left(\mathbf{w}_t\right) = \frac{1}{K}\sum_{k=1}^{K} \left\lVert \mathbf{w}_t^{(k)} - \overline{\mathbf{w}}_t \right\rVert_2^2 \tag{2}$$

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 or low variance suggests that the workers' models are closely aligned, working collectively towards the shared objective.

The FDA algorithm (Algorithm 1) is based on the premise that, as long as the variance is below a threshold Θ, synchronization is not needed. Thus, we introduce the *Round Invariant* (RI):

$$\mathcal{V}ar\left(\mathbf{w}_t\right) \leq \Theta \tag{3}$$

To preserve the RI, our FDA algorithm maintains (Lines 4-6 of Algorithm 1) at each worker k a local (low-dimensional) state-vector $\mathrm{S}_t^{(k)}$, which is computed based on $\mathbf{w}_t^{(k)}$. These state vectors are vital for the subsequent estimation of the model variance, and underpin the two variants of the FDA algorithm (provided in Sections 3.1 and 3.2, respectively). Our estimation techniques begin by performing AllReduce on the states $\mathrm{S}_t^{(k)}$, consolidating them into the average state $\overline{\mathrm{S}}_t$ (Line 7). Importantly, this communication step requires significantly less bandwidth and resources than transmitting the full models $\mathbf{w}_t^{(k)}$.

For each FDA variant, we also define a (different) function $H(\overline{\mathrm{S}}_t)$ that overestimates the variance, i.e., it ensures that as long as $H(\overline{\mathrm{S}}_t) \leq \Theta$ then the variance is bounded by Θ. This guarantee is probabilistic for the Sketch-based variant of FDA, and deterministic for its Linear counterpart. Consequently, if $H(\overline{\mathrm{S}}_t) > \Theta$ then synchronization is performed (Lines 8-9) — the RI invariant cannot be guaranteed. After synchronization, the model variance is zero.

Efficiently Monitoring the RI. Estimating model variance efficiently is at the heart of FDA. To this end, we first introduce the *local model drift*, $\mathbf{u}_t^{(k)}$, and *average drift*, $\overline{\mathbf{u}}_t$, defined as follows:

$$\mathbf{u}_t^{(k)} = \mathbf{w}_t^{(k)} - \overline{\mathbf{w}}_{t_0} \quad , \quad \overline{\mathbf{u}}_t = \frac{1}{K}\sum_{k=1}^{K} \mathbf{u}_t^{(k)}$$

Here, $\overline{\mathbf{w}}_{t_0}$ denotes the model vector after the most recent synchronization. Subsequently, the model variance can be written as:

$$\mathcal{V}ar\left(\mathbf{w}_t\right) = \left(\frac{1}{K}\sum_{k=1}^{K} \left\lVert \mathbf{u}_t^{(k)} \right\rVert_2^2\right) - \lVert \overline{\mathbf{u}}_t \rVert_2^2 \tag{4}$$

Proof. Adding an offset $(-\overline{\mathbf{w}}_{t_0})$ to each $\mathbf{w}_t^{(k)}$ does not alter the variance, therefore:

$$\begin{aligned}
\mathcal{V}ar\left(\mathbf{w}_t\right) &= \mathcal{V}ar\left(\mathbf{w}_t - \overline{\mathbf{w}}_{t_0}\right) = \mathcal{V}ar\left(\mathbf{u}_t\right) = \frac{1}{K}\sum_{k=1}^{K} \left\lVert \mathbf{u}_t^{(k)} - \overline{\mathbf{u}}_t \right\rVert_2^2 \\
&= \frac{1}{K}\sum_{k=1}^{K} \left(\left\lVert \mathbf{u}_t^{(k)} \right\rVert_2^2 - 2\left\langle \mathbf{u}_t^{(k)}, \overline{\mathbf{u}}_t \right\rangle + \lVert \overline{\mathbf{u}}_t \rVert_2^2 \right) \\
&= \left(\frac{1}{K}\sum_{k=1}^{K} \left\lVert \mathbf{u}_t^{(k)} \right\rVert_2^2\right) - 2\left(\frac{1}{K}\sum_{k=1}^{K} \left\langle \mathbf{u}_t^{(k)}, \overline{\mathbf{u}}_t \right\rangle\right) + \left(\frac{1}{K}\sum_{k=1}^{K} \lVert \overline{\mathbf{u}}_t \rVert_2^2\right) \\
&= \left(\frac{1}{K}\sum_{k=1}^{K} \left\lVert \mathbf{u}_t^{(k)} \right\rVert_2^2\right) - 2\left\langle \left(\frac{1}{K}\sum_{k=1}^{K} \mathbf{u}_t^{(k)}\right), \overline{\mathbf{u}}_t \right\rangle + \lVert \overline{\mathbf{u}}_t \rVert_2^2 \\
&= \left(\frac{1}{K}\sum_{k=1}^{K} \left\lVert \mathbf{u}_t^{(k)} \right\rVert_2^2\right) - 2\left\langle \overline{\mathbf{u}}_t, \overline{\mathbf{u}}_t \right\rangle + \lVert \overline{\mathbf{u}}_t \rVert_2^2 \\
&= \left(\frac{1}{K}\sum_{k=1}^{K} \left\lVert \mathbf{u}_t^{(k)} \right\rVert_2^2\right) - 2\lVert \overline{\mathbf{u}}_t \rVert_2^2 + \lVert \overline{\mathbf{u}}_t \rVert_2^2 \\
&= \left(\frac{1}{K}\sum_{k=1}^{K} \left\lVert \mathbf{u}_t^{(k)} \right\rVert_2^2\right) - \lVert \overline{\mathbf{u}}_t \rVert_2^2
\end{aligned}$$

□

Algorithm 1 Federated Dynamic Averaging - FDA

```
Require: $K$: The number of workers indexed by $k$
Require: $\Theta$: The model variance threshold
Require: $b$: The local mini-batch size

1: Initialize $\mathbf{w}_0^{(k)} = \overline{\mathbf{w}}_0 \in \mathbb{R}^d$
2: for each step $t = 1, 2, \ldots$ do
3:     for each worker $k = 1, \ldots, K$ in parallel do
4:         $\mathcal{B}_t^{(k)} \leftarrow$ (sample a batch of size $b$ from $\mathcal{D}_k$)
5:         $\mathbf{w}_t^{(k)} \leftarrow$ OPTIMIZE($\mathbf{w}_{t-1}^{(k)}, \mathcal{B}_t^{(k)}$)
6:         Update $\mathrm{S}_t^{(k)}$
7:         $\overline{\mathrm{S}}_t \leftarrow$ ALLREDUCE($\mathrm{S}_t^{(k)}$)
8:         if $H(\overline{\mathrm{S}}_t) > \Theta$ then
9:             $\mathbf{w}_t^{(k)} \leftarrow$ ALLREDUCE($\mathbf{w}_t^{(k)}$)      ▷ In-place
```

Conceptually, following Eq (4), to precisely monitor the variance, we need to calculate two quantities: (1) $\frac{1}{K}\sum_{k=1}^{K} \|\mathbf{u}_t^{(k)}\|_2^2$, and (2) $\|\overline{\mathbf{u}}_t\|_2^2$. The first quantity requires an ALLREDUCE operation on the squared norm of the worker drifts, which involves minimal overhead since these values are scalar. In contrast, the second quantity necessitates an ALLREDUCE operation on the worker drifts themselves, which are of model dimension, thus incurring a high communication cost. In fact, this operation is equivalent to synchronization, which is exactly what we aim to avoid in the first place. Thus, it becomes evident that communication-efficient model variance estimation hinges on estimating $\|\overline{\mathbf{u}}_t\|_2^2$ efficiently.

Upcoming sections will detail two techniques for communication efficient variance estimation (which primarily involves estimating $\|\overline{\mathbf{u}}_t\|_2^2$): SKETCHFDA and LINEARFDA. To present them uniformly, we introduce the *local state* $\mathrm{S}_t^{(k)}$, a tensor which contains: (1) the scalar value $\|\mathbf{u}_t^{(k)}\|_2^2$ for precisely calculating the first quantity, and (2) a low-dimensional summary of $\mathbf{u}_t^{(k)}$, different for each technique, for estimating the second quantity. For each technique we define an estimation function $H(\cdot)$ that calculates the current variance estimate from *average state* $\overline{\mathrm{S}}_t = \frac{1}{K}\sum_{k=1}^{K} \mathrm{S}_t^{(k)}$ (obtained via ALLREDUCE).

3.1 SKETCHFDA: Sketch-based Estimation

An optimal estimator for $\|\overline{\mathbf{u}}_t\|_2^2$ can be obtained through the utilization and properties of AMS sketches, as detailed in [8]. An AMS sketch of a vector $\mathbf{v} \in \mathbb{R}^d$ is an $l \times m$ real matrix:

$$\mathrm{sk}(\mathbf{v}) = \begin{bmatrix} \psi_1 & \psi_2 & \ldots & \psi_l \end{bmatrix}^\top \in \mathbb{R}^{l \times m} \quad , \quad l \cdot m \ll d$$

An estimate for squared-norm $\|\mathbf{v}\|_2^2$ is provided by the formula

$$\mathcal{M}_2(\mathrm{sk}(\mathbf{v})) = \mathrm{median}\left\{\|\psi_i\|_2^2 \ , \ i = 1, \ldots, l\right\}$$

The quality of estimation depends on the size of the sketch. For chosen $\epsilon, \delta > 0$, where sketch dimensions are given by $l = O(\log 1/\delta)$ and $m = O(1/\epsilon^2)$, we have the following probabilistic guarantee: with confidence at least $1 - \delta$,

$$\mathcal{M}_2(\mathrm{sk}(\mathbf{v})) \in (1 \pm \epsilon)\, \|\mathbf{v}\|_2^2$$

Notably, observe that the accuracy (ϵ) and confidence ($1 - \delta$) only depend on the size of the sketch and not on the dimensionality of vector $\mathbf{v}$.

Two crucial properties of the AMS sketch are that (a) it is a linear transformation, i.e., for $\alpha_1, \alpha_2 \in \mathbb{R}$ and $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^d$,

$$\mathrm{sk}(\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2) = \alpha_1\,\mathrm{sk}(\mathbf{v}_1) + \alpha_2\,\mathrm{sk}(\mathbf{v}_2)$$

and (b) can be computed efficiently in time $O(l \cdot d)$.

In the SKETCHFDA approach, the salient idea is to employ AMS sketches $\mathrm{sk}(\mathbf{u}_t^{(k)}) \in \mathbb{R}^{l \times m}$ as a low-dimensional representation of the local drifts $\mathbf{u}_t^{(k)}$.

THEOREM 3.1. *Let $l = O(\log\frac{1}{\delta})$ and $m = O(\frac{1}{\epsilon^2})$. Define the local state as*

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

and the approximation function as

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

Then, the condition $H(\overline{\mathrm{S}}_t) \le \Theta$ implies $\mathcal{V}ar(\mathbf{w}_t) \le \Theta$ with probability at least $(1 - \delta)$.

PROOF.

$$\begin{aligned} H\left(\overline{\mathrm{S}}_t\right) &= \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \frac{1}{1+\epsilon}\mathcal{M}_2\left(\frac{1}{K}\sum_{i=1}^{K}\mathrm{sk}\left(\mathbf{u}_t^{(k)}\right)\right) \\ &\overset{\text{(lin.)}}{=} \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \frac{1}{1+\epsilon}\mathcal{M}_2\left(\mathrm{sk}\left(\frac{1}{K}\sum_{i=1}^{K}\mathbf{u}_t^{(k)}\right)\right) \\ &= \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \frac{1}{1+\epsilon}\mathcal{M}_2\left(\mathrm{sk}\left(\overline{\mathbf{u}}_t\right)\right) \\ &\overset{(\epsilon\text{-err.})}{\ge} \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \|\overline{\mathbf{u}}_t\|_2^2 \quad \text{with prob. at least } (1-\delta) \\ &= \mathcal{V}ar\left(\mathbf{w}_t\right) \end{aligned}$$

We proved that $H(\overline{\mathrm{S}}_t) \ge \mathcal{V}ar(\mathbf{w}_t)$ with probability at least $(1-\delta)$, i.e., we overestimate the model variance with probability at least $(1 - \delta)$, completing the proof. □

In Section 3.3, we discuss the empirical basis for choosing the values of l and m, and how they practically impact the quality of the sketch approximation.

3.2 LINEARFDA: Linear Approximation

Although AMS sketches provide good estimates for variance, their dimension is in the several hundreds, and the communication cost of ALLREDUCE on sketches, performed at each step, may be non-negligible. Therefore, we also introduce a low-cost, ad-hoc estimation variant.

In this approach, instead of an AMS sketch, each local state contains the scalar value $\langle \xi, \mathbf{u}_t^{(k)} \rangle \in \mathbb{R}$, where $\xi \in \mathbb{R}^d$ is a unit vector, known to all workers.

THEOREM 3.2. *Define the local state as*

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

and the approximation function as

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

Then, the condition $H(\overline{\mathrm{S}}_t) \le \Theta$ implies $\mathcal{V}ar(\mathbf{w}_t) \le \Theta$.

Proof.

$$\begin{aligned}
H\left(\overline{\mathrm{S}}_t\right) &= \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \left|\frac{1}{K}\sum_{i=1}^{K}\left\langle \xi,\, \mathbf{u}_t^{(k)}\right\rangle\right|^2 \\
&= \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \left|\left\langle \xi,\, \frac{1}{K}\sum_{i=1}^{K}\mathbf{u}_t^{(k)}\right\rangle\right|^2 \\
&= \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - |\langle \xi,\, \overline{\mathbf{u}}_t\rangle|^2 \\
&\geq \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \|\xi\|_2^2\,\|\overline{\mathbf{u}}_t\|_2^2 \\
&= \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{u}_t^{(k)}\right\|_2^2 - \|\overline{\mathbf{u}}_t\|_2^2 \\
&= \mathcal{V}ar\,(\mathbf{w}_t)
\end{aligned}$$

We proved that $H(\overline{\mathrm{S}}_t) \geq \mathcal{V}ar\,(\mathbf{w}_t)$, i.e., we always overestimate the model variance, completing the proof. □

An arbitrary choice of ξ (e.g., a random vector) is likely to estimate $\|\overline{\mathbf{u}}_t\|_2^2$ poorly; if ξ is uncorrelated to $\overline{\mathbf{u}}_t$, then $|\langle \xi,\, \overline{\mathbf{u}}_t\rangle|^2$ will likely be close to zero. A heuristic choice that might be correlated to $\overline{\mathbf{u}}_t$ is the (normalized) value of $\overline{\mathbf{u}}_{t_0}$, the global drift vector right at the time of last synchronization. All nodes can compute it independently without extra communication, if they take the difference of the models of the last two synchronizations:

$$\xi = \frac{\overline{\mathbf{u}}_{t_0}}{\left\|\overline{\mathbf{u}}_{t_0}\right\|_2} = \frac{\overline{\mathbf{w}}_{t_0} - \overline{\mathbf{w}}_{t_{-1}}}{\left\|\overline{\mathbf{w}}_{t_0} - \overline{\mathbf{w}}_{t_{-1}}\right\|_2}$$

3.3 Discussion

FDA: Intuition. The main intuition for FDA is summarized in making the decision to synchronize dynamic, based on *model variance* during training. This metric is designed to capture the collective state of the training process. In what follows, we provide intuition on why this is the case. It is important to remember that the global model $\overline{\mathbf{w}}_t$ and, by extension, the global drift $\overline{\mathbf{u}}_t$, are ultimately what we care about and evaluate.

Model variance, as defined in Equation (4), is the difference between the average of the squared local drifts $\frac{1}{K}\sum\|\mathbf{u}_t^{(k)}\|_2^2$ and the squared global drift $\|\overline{\mathbf{u}}_t\|_2^2$. The first term reflects how far the individual worker models have moved–essentially, how much each worker has learned. The second term indicates how much of this learning is retained in the global model after aggregation.

The interplay between these two quantities is crucial. For example, when the local drifts are high but the global drift is low, the variance increases, signaling the need for synchronization. This scenario suggests that while individual workers have made significant progress (as indicated by high local drifts), this progress is not being effectively captured in the global model (indicated by the low global drift). In other words, the worker models have moved significantly, but the global model has remained relatively stationary in this high-dimensional space. This misalignment indicates that training is no longer progressing optimally, as the workers are moving towards disparate and conflicting local minima, making it crucial to synchronize and realign them. Conversely, when both the local and global drifts are either low or high, synchronization is not necessary, and the variance naturally remains low.

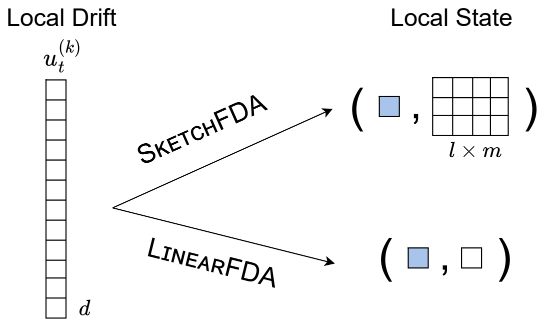


Figure 2: SketchFDA & LinearFDA: Local State structure.

Neither the average of the local drifts nor the global drift alone provides a complete picture of the collective training progress. Relying solely on one or the other would lead to suboptimal synchronization decisions and likely prove ineffective. In FDA, it is the relationship between these quantities, as captured by the model variance, that offers valuable insights and guides the crucial decision of when to synchronize.

SketchFDA vs. LinearFDA: Both methods send the squared norm of the drift $\|\mathbf{u}_t^{(k)}\|_2^2$, but differ in the additional accompanying lower-dimensional representation they transmit (Figure 2):

(1) SketchFDA: An AMS sketch of the local drift.
(2) LinearFDA: The dot product of a vector and the local drift.

The key difference between these two variants lies in the fidelity of approximation of the model variance. While both methods conservatively overestimate the variance, SketchFDA provides a provably accurate estimation, which is expected to lead to fewer synchronizations. LinearFDA requires less computational effort and bandwidth to create and communicate the local states, but may overestimate variance by too much, causing unnecessary synchronizations.

SketchFDA: Choice of l and m. We empirically measured the approximation achieved with sketch dimensions of $l = 5$ rows and $m = 250$ columns (as defined in Section 3.1): these settings yield an error bound of $\epsilon \approx 6\%$ and a probabilistic confidence of $(1 - \delta) \approx 95\%$. Based on our experiments, we have adopted these values in our experiments and recommend them. Using these values, the byte-size of a sketch is $l \cdot m \cdot 4\,\text{bytes} = 5\,\text{kB}$, significantly smaller than the size of all our models. Sketches of smaller size could be used, albeit weakening the approximation of the variance. However, given that LinearFDA similarly weakens approximation and avoids using AMS sketches, in the interest of space we do not explore varying AMS sketch sizes in this paper.

FDA: Asynchronous Operation. As mentioned in Section 2, FDA can be readily modified to operate asynchronously. In this setup, one worker-node acts as a coordinator, aggregating local states and determining whether synchronization is needed each time a local state is received. This decision is based on the most recent local states from all workers. It is important to note that, since local states are small in size, asynchronous operation is unlikely to alleviate bandwidth issues. The primary advantage is that it allows training to continue even in the presence of stragglers. Asynchronous operation might also be beneficial in rare cases where the overhead of initializing communication dominates the actual transmission time.

4 EXPERIMENTS

4.1 Setup

Table 2 provides a comprehensive overview of our experiments. For each experiment, we detail the Neural Network (NN) architecture, its parameter count (d), and the dataset used for training. The table also specifies key hyper-parameters: the batch size (b), the number of workers (K), and the FDA-specific variance threshold (Θ). Additionally, we indicate the chosen optimizer (as detailed in Section 3) and the training algorithms employed for each configuration.

Platform. We employ TensorFlow [1], integrated with Keras [7], as the platform for conducting our experiments. We used TensorFlow to implement our FDA variants and all competitive algorithms. All relevant code, figures, and data of this study are available in https://github.com/miketheologitis/FedL-Sync-FDA.

Hardware & Infrastructure. We conducted our experiments on the ARIS High performance computing (HPC) environment[2], utilizing a cluster of 44 GPU-accelerated worker-nodes. Each worker is equipped with two NVIDIA Tesla K40m GPUs and interconnected via an InfiniBand FDR14 network, providing up to 56 GB/s of bandwidth. Crucially, our evaluation remains agnostic to the underlying infrastructure of the specific workers.

Datasets & Models. The core experiments involve training Convolutional Neural Networks (CNNs) of varying sizes and complexities on two datasets: MNIST [12] and CIFAR-10 [27]. For the MNIST dataset, we employ LeNet-5 [29], composed of approximately 62 thousand parameters, and a modified version of VGG16 [48], 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 [44], 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 [15]. For CIFAR-10, we utilize DenseNet121 and DenseNet201 [22], as implemented in Keras [7], with the addition of dropout regularization layers at rate 0.2 and weight decay of 10^{-4}, as prescribed in [22]. The DenseNet121 and DenseNet201 models have 6.9 million and 18 million parameters, respectively, and are both initialized with He normal [19].

Lastly, we explore a transfer learning scenario on the dataset CIFAR-100 [27], a choice reflecting the DL community's growing preference of using pre-trained models in such downstream tasks [18]. 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 [13]. We adopt this exact transfer learning scenario, leveraging the more powerful ConvNeXtLarge model, pre-trained on ImageNet, with 198 million parameters [7, 33]. Following the feature extraction step [16], 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 [39].

Algorithms. We consider five distributed deep learning algorithms: LinearFDA, SketchFDA, Synchronous [3], FedAdam [42], and FedAvgM [21]; the first three are standard in all experiments. Depending on the local optimizer, Adam [26] or SGD with Nesterov momentum (SGD-NM) [52], we also include their communication-efficient 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 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. Notably, communication cost is unaffected by the training data volume since only model updates (when synchronizing) and local states (at each step), but not training data, are transmitted. Thus, the communication cost mainly depends on the complexity (number of parameters) of the used model. Translating the communication 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 and on the overhead of establishing and initializing communication. 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 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 worker-nodes. Its impact is larger for workers with lower computational resources.

Hyper-Parameters & Optimizers. Hyper-parameters unique to each training dataset and model are detailed in Table 2; Θ is pertinent to FDA algorithms and not applicable to others. Notably, a guideline for setting the parameter Θ is provided in Section 4.3. For experiments involving FedAvgM and FedAdam, we use $E = 1$ local epochs, following [42]. For experiments with LeNet-5 and VGG16*, local optimization employs Adam, using the default settings as per [26]. In these cases, FedAdam also adheres to the default settings for both local and server optimization [7, 42]. 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 [22]. For FedAvgM, local optimization is conducted with default settings [7, 21], while server optimization employs SGD with momentum, setting the momentum parameter and learning rate to 0.9 and 0.316, respectively [42]. Lastly, for the transfer learning experiments, local optimization leverages AdamW [34], with the hyper-parameters used for fine-tuning ConvNeXtLarge in the original study [33].

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:

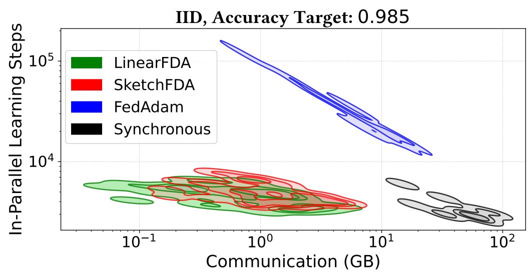
(1) **IID** — Independent and identically distributed.
(2) **Non-IID:** $X\%$ — A portion $X\%$ of the dataset is sorted by label and sequentially allocated to workers, with the remainder distributed in an IID fashion.

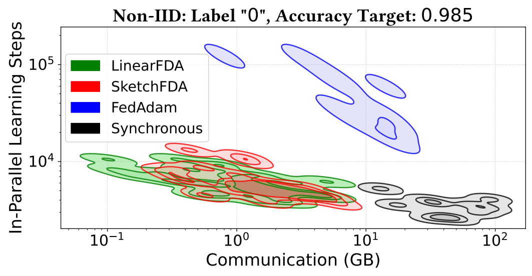
[2]https://www.hpc.grnet.gr/en/hardware-2/

[3]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 2: Summary of Experiments

			Hyper-Parameters			Training	
NN	d	Dataset	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





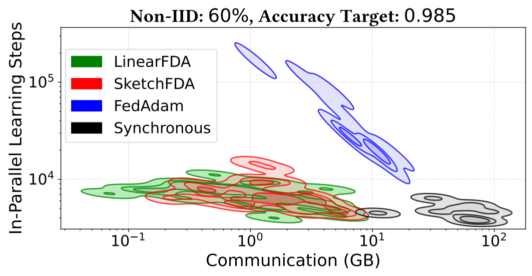


Figure 3: LeNet-5 on MNIST. At Non-IID: Label "0", the samples of Label "0" are assigned to few workers. At Non-IID: 60%, 60% of the dataset is sorted and allocated to workers, causing some workers to receive many samples from the same label

(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.

4.2 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 two numbers). 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.

4.3 Results

Due to the extensive set of unique experiments (over 1000), as detailed in Table 2, we leverage Kernel Density Estimation (KDE) plots [62] 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. The varying levels of opacity in the filled areas of the KDE plots represent the density of the underlying data points: higher opacity indicates areas with a greater concentration of data, whereas lower opacity signifies less dense areas.

As an illustrative example, Figure 3 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 2) that attained the *Accuracy Target* of 0.985. The observed high variance in the method's distribution stems from the varying K and Θ values. In subsequent subsections, we elucidate how these hyper-parameters influence the communication and computation costs.

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 3, 4, 5, and 6 (where Synchronous appears in the bottom right — low computation, very high communication). Conversely, Federated Optimization (FedOpt) methods [42] are designed to be communication-efficient, reducing communication between

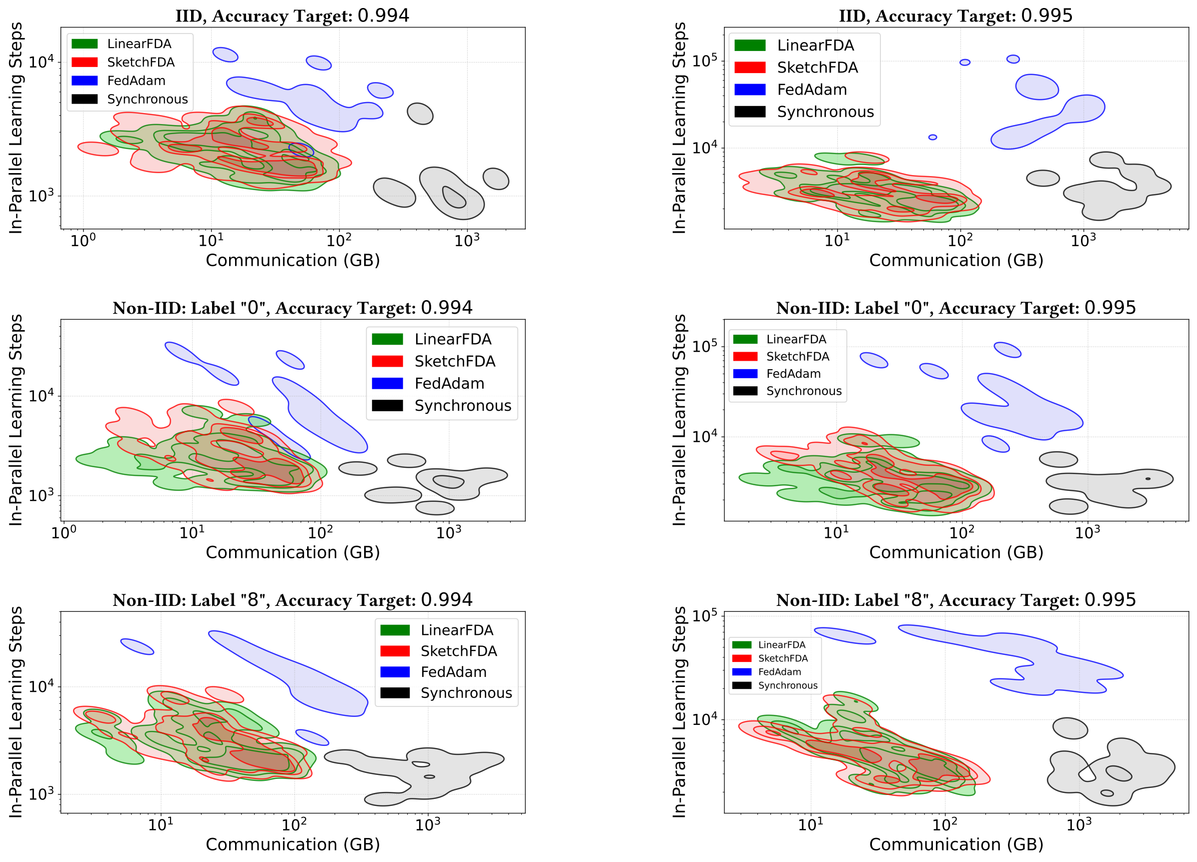


Figure 4: VGG16* on MNIST

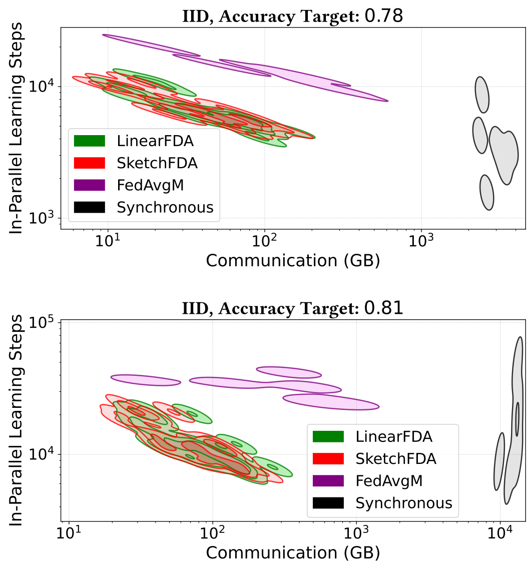


Figure 5: DenseNet121 on CIFAR-10

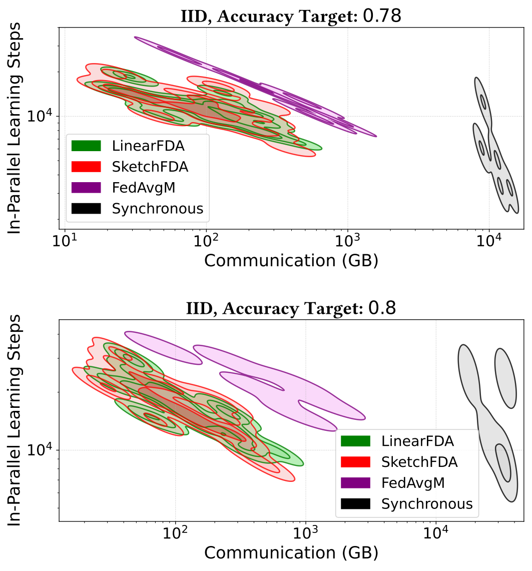


Figure 6: DenseNet201 on CIFAR-10

devices (workers) at the expense of increased local computation. Indeed, as shown in Figures 3-6, 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 3, 4, 5, and 6. In fact, they significantly outperform FedAvgM and FedAdam in their element, that is, communication-efficiency. Across all experiments,

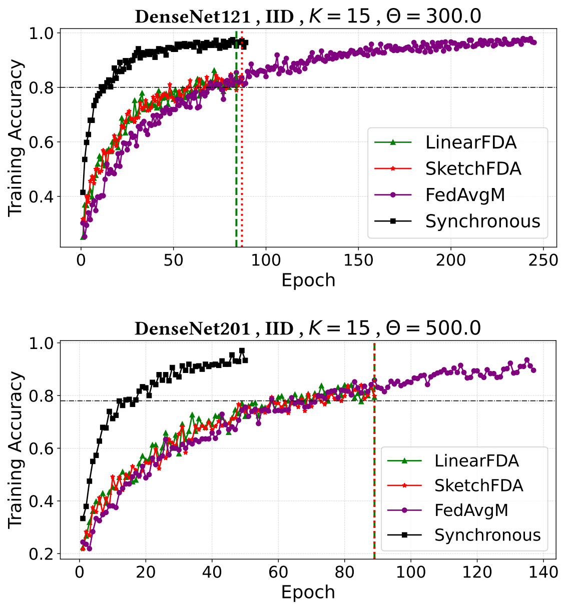


Figure 7: Training accuracy progression with a test accuracy target (horizontal line) of 0.8 (top), and 0.78 (bottom). Dashed and doted lines indicate when LinearFDA and SketchFDA attain the target accuracy, respectively. A smaller final gap between training and target accuracy indicates less overfitting, i.e., better generalization capabilities of the trained model

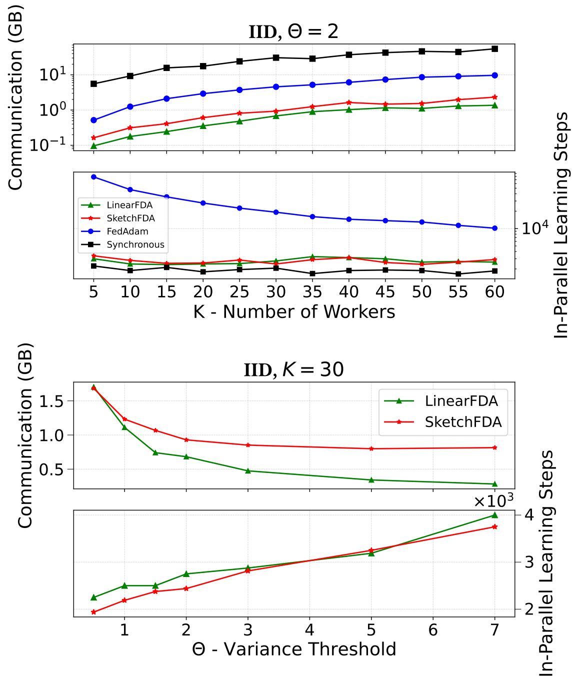


Figure 8: LeNet-5 on MNIST: Varying the Number of Workers and Θ — Accuracy Target: 0.98

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 [16, 54]. We first clearly notice this with VGG16* on MNIST in Figure 4 for all three data heterogeneity settings. For a 0.001 increase in accuracy (effectively 10 misclassified testing images), FedAdam needs approximately 2-7× more communication and 3-7× more computation, respectively. This can be seen by comparing the figures at the left column of Figure 4 with the corresponding ones in the right column. 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 5, and 6), 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 3, 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 4 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 has a lower generalization gap. 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 DL: underfitting and overfitting [16]. For DenseNet121 on CIFAR-10, with a test accuracy target of 0.8, as illustrated in Figure 7, 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 7. Notably, given the necessity to fix hyper-parameters 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.

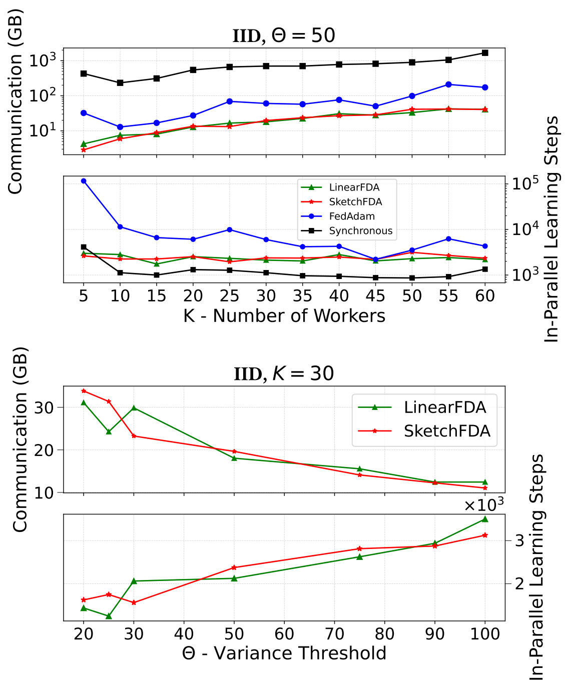


Figure 9: VGG16* on MNIST: Varying the Number of Workers and Θ — Accuracy Target: 0.994

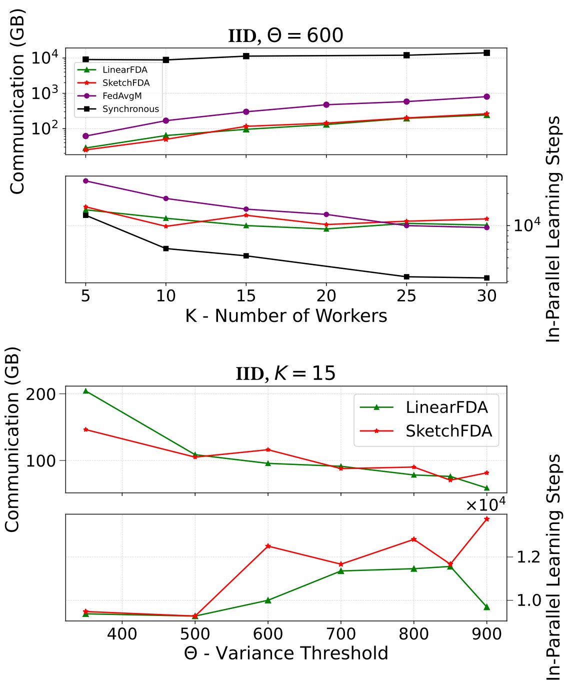


Figure 11: DenseNet201 on CIFAR-10: Varying the Number of Workers and Θ — Accuracy Target: 0.78

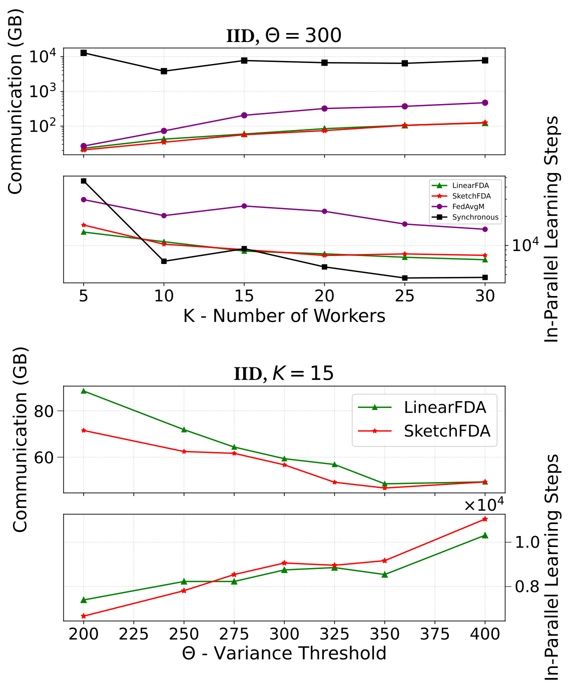


Figure 10: DenseNet121 on CIFAR-10: Varying the Number of Workers and Θ — Accuracy Target: 0.8

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 [66]. This unpredictability is starkly illustrated with LeNet-5 and VGG16* on the MNIST dataset across all data heterogeneity settings and all strategies. Figures 8, and 9 (top) demonstrate that increasing the number of workers does not decrease computation – except for FedAdam which begins with significantly high computation – but rather exacerbates communication. 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 (top part of Figures 10, 11), the expected behavior starts to emerge. Especially for DenseNet-121, scaling up (K increase) leads to a 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. Our FDA variants consistently outperform FedAdam, FedAvgM, and Synchronous in communication efficiency, as demonstrated across all experiments in Figures 8-11. Specifically, they require up to 30 times less communication than FedAdam, 4 times less than FedAvgM, and up to 2.5 orders of magnitude less than Synchronous.

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 both FDA strategies, and all learning tasks, and data heterogeneity settings (Figures 8-11). Interestingly, for more complex models like DenseNet121 and DenseNet201 on CIFAR-10,

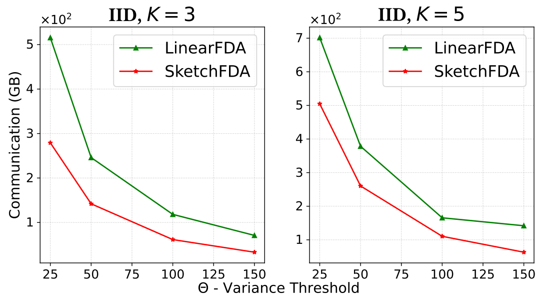


Figure 13: ConvNeXtLarge on CIFAR-100 (transfer learning from ImageNet) — Deployment of FDA during the fine-tuning stage with Accuracy Target of 0.76

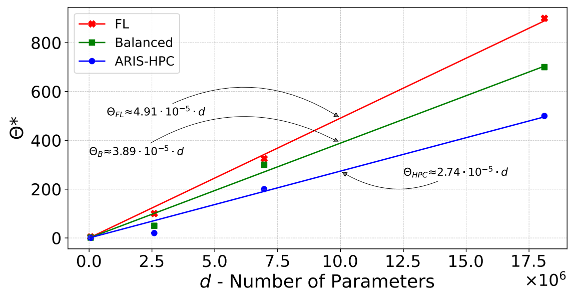


Figure 12: Empirical Estimation of the Variance Threshold

increasing the variance threshold (Θ) does not lead to a significant rise in computation cost, as illustrated in Figures 10 and 11. 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 8, 9, 10, and 11. 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 2), 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 the 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 the variance threshold, Figure 12 presents empirical estimations for Θ across three distinct learning settings:

(1) FL, assuming a common channel of 0.5Gbps, where

$$\Theta_{FL} = 4.91 \cdot 10^{-5} \cdot d$$

(2) Balanced communication-computation equilibrium, where

$$\Theta_B = 3.89 \cdot 10^{-5} \cdot d$$

(3) Our HPC environment at the ARIS supercomputer, where

$$\Theta_{HPC} = 2.74 \cdot 10^{-5} \cdot d$$

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 3, 4, 5, and 6. This suggests that the precision of the variance approximation is not critical. 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 ConvNeXtLarge model to equivalent performance levels (Figure 13). 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 computation. On the other hand, for intricate learning tasks and deeper models, SketchFDA becomes the preferred choice, if communication-efficiency is paramount.

5 CONCLUSIONS AND FUTURE WORK

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 synchronize the local models 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 over-fitting. Our results push the limits of modern communication-efficient distributed deep learning, paving the way for more scalable, dynamic, and broadly applicable strategies.

An interesting direction for future work is whether the value of Θ can be dynamically adjusted in order to achieve (or not to exceed) a target average bandwidth consumption. Since the expected behavior is that the communication cost decreases when Θ increases, such an approach seems feasible (i.e., increasing Θ when the bandwidth consumption is higher than what is desired), especially by using statistics. We plan to look into this extension in the future.

ACKNOWLEDGMENTS

We wish to thank the anonymous reviewers and the meta-reviewer for their insightful comments and suggestions. This work was supported by the EU project CREXDATA under Horizon Europe agreement No. 101092749. Moreover, this work was supported by computational time granted from the National Infrastructures for Research and Technology S.A. (GRNET S.A.) in the National HPC facility - ARIS - under project ID pa230902-fda1.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16)*. USENIX Association, USA, 265–283.

[2] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. 2021. Federated Learning Based on Dynamic Regularization. In *International Conference on Learning Representations*.

[3] Alham Fikri Aji and Kenneth Heafield. 2017. Sparse Communication for Distributed Gradient Descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Martha Palmer, Rebecca Hwa, and Sebastian Riedel (Eds.). Association for Computational Linguistics, Copenhagen, Denmark, 440–445.

[4] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. 2019. *Qsparse-local-SGD: distributed SGD with quantization, sparsification, and local computations*. Curran Associates Inc., Red Hook, NY, USA.

[5] Tianyi Chen, Georgios B. Giannakis, Tao Sun, and Wotao Yin. 2018. LAG: lazily aggregated gradient for communication-efficient distributed learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 5055–5065.

[6] 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.

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

[8] 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 (VLDB '05)*. VLDB Endowment, 13–24.

[9] Graham Cormode, Igor L. Markov, and Harish Srinivas. 2024. Private and Efficient Federated Numerical Aggregation. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, Letizia Tanca, Qiong Luo 0001, Giuseppe Polese, Loredana Caruccio, Xavier Oriol, and Donatella Firmani (Eds.). 734–742.

[10] Angjela Davitkova, Damjan Gjurovski, and Sebastian Michel 0001. 2024. Learning over Sets for Databases. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, Letizia Tanca, Qiong Luo 0001, Giuseppe Polese, Loredana Caruccio, Xavier Oriol, and Donatella Firmani (Eds.). OpenProceedings.org, 68–80.

[11] 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.

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

[13] 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. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[14] Fangcheng Fu, Xupeng Miao, Jiawei Jiang, Huanran Xue, and Bin Cui. 2022. Towards communication-efficient vertical federated learning training via cache-enabled local updates. 15, 10 (jun 2022), 2111–2120.

[15] 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.

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

[17] Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck R. Cadambe. 2019. Local SGD with Periodic Averaging: Tighter Analysis and Adaptive Synchronization. In *Neural Information Processing Systems*.

[18] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, 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. *AI Open* 2 (2021), 225–250.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 1026–1034.

[20] Elad Hoffer, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 1729–1739.

[21] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. *CoRR* (2019).

[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 2261–2269.

[23] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, 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. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends in Machine Learning* 14, 1–2 (jun 2021), 1–210.

[24] 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.

[25] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 5132–5143.

[26] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. San Diego, CA, USA.

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

[28] Eugenie Yujing Lai, Zainab Zolaktaf, Mostafa Milani, Omar AlOmeir, Jianhao Cao, and Rachel Pottinger. 2023. Workload-Aware Query Recommendation Using Deep Learning. In *Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023*, Julia Stoyanovich, Jens Teubner, Nikos Mamoulis, Evaggelia Pitoura, and Jan Mühlig (Eds.). OpenProceedings.org, 53–65.

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

[30] 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. *Proc. VLDB Endow.* 13, 12 (aug 2020), 3005–3018.

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

[32] Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. 2020. Don't Use Large Mini-batches, Use Local SGD. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[33] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. 2022. A ConvNet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 11966–11976.

[34] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

[35] Kaihao Ma, Xiao Yan, Zhenkun Cai, Yuzhen Huang, Yidi Wu, and James Cheng. 2023. FEC: Efficient Deep Recommendation Model Training with Flexible Embedding Communication. *Proc. ACM Manag. Data* 1, 2, Article 165 (jun 2023), 21 pages.

[36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Aarti Singh and Jerry Zhu (Eds.), Vol. 54. PMLR, 1273–1282.

[37] Xupeng Miao, Xiaonan Nie, Yingxia Shao, Zhi Yang, Jiawei Jiang, Lingxiao Ma, and Bin Cui. 2021. Heterogeneity-Aware Distributed Machine Learning Training via Partial Reduce. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 2262–2270.

[38] Jed Mills, Jia Hu, and Geyong Min. 2023. Faster Federated Learning With Decaying Number of Local SGD Steps. *IEEE Trans. Parallel Distrib. Syst.* 34, 7 (jul 2023), 2198–2207.

[39] Supun Nakandala and Arun Kumar. 2022. Nautilus: An Optimized System for Deep Transfer Learning over Evolving Training Datasets. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 506–520. https://doi.org/10.1145/3514221.3517846

[40] 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.

[41] Hafiz Tayyab Rauf, André Freitas, and Norman W. Paton. 2024. Deep Clustering for Data Cleaning and Integration. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, Letizia Tanca, Qiong Luo 0001, Giuseppe Polese, Loredana Caruccio, Xavier Oriol, and Donatella Firmani (Eds.). OpenProceedings.org, 636–649. https://doi.org/10.48786/edbt.2024.55

[42] Sashank Reddi, Zachary Burr Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Brendan McMahan (Eds.). 2021. *Adaptive Federated Optimization.*

[43] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. 2020. FetchSGD: communication-efficient federated learning with sketching. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20).* JMLR.org, Article 764, 13 pages.

[44] 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. *International Journal of Computer Vision* 115, 3 (01 Dec 2015), 211–252.

[45] Anit Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. 2018. On the Convergence of Federated Optimization in Heterogeneous Networks. (12 2018).

[46] 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.

[47] Nir Shlezinger, Mingzhe Chen, Yonina Eldar, H. Vincent Poor, and Shuguang Cui. 2021. UVeQFed: Universal Vector Quantization for Federated Learning. *IEEE Transactions on Signal Processing* 69 (01 2021), 500–514.

[48] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[49] Ryan Spring, Anastasios Kyrillidis, Vijai Mohan, and Anshumali Shrivastava. 2019. Compressing Gradient Optimizers via Count-Sketches. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 5946–5955.

[50] Sebastian U. Stich. 2019. Local SGD Converges Fast and Communicates Little. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.* OpenReview.net.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] Taegeon Um, Byungsoo Oh, Byeongchan Seo, Minhyeok Kweun, Goeun Kim, and Woo-Yeon Lee. 2023. FastFlow: Accelerating Deep Learning Model Training with Smart Offloading of Input Data Pipeline. 16, 5 (jan 2023), 1086–1099.

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

[57] Jianyu Wang and Gauri Joshi. [n.d.]. Adaptive Communication Strategies to Achieve the Best Error-Runtime Trade-off in Local-update SGD. *Systems and Machine Learning (SysML) Conference* ([n. d.]).

[58] 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.

[59] 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.

[60] Wei Wang, Meihui Zhang, Gang Chen, H. V. Jagadish, Beng Chin Ooi, and Kian-Lee Tan. 2016. Database Meets Deep Learning: Challenges and Opportunities. 45, 2 (sep 2016), 17–22.

[61] 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.

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

[63] Phillip Wenig and Thorsten Papenbrock. 2022. DataGossip: A Data Exchange Extension for Distributed Machine Learning Algorithms. In *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*, Julia Stoyanovich, Jens Teubner, Paolo Guagliardo, Milos Nikolic, Andreas Pieris, Jan Mühlig, Fatma Özcan, Sebastian Schelter, H. V. Jagadish, and Meihui Zhang 0001 (Eds.).

[64] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. 2020. Privacy preserving vertical federated learning for tree-based models. *Proc. VLDB Endow.* 13, 12 (jul 2020), 2090–2103.

[65] Hao Yu and Rong Jin. 2019. On the Computation and Communication Complexity of Parallel SGD with Dynamic Batch Sizes for Stochastic Non-Convex Optimization. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 7174–7183.

[66] Hao Yu, Sen Yang, and Shenghuo Zhu. 2019. Parallel restarted SGD with faster convergence and less communication: demystifying why model averaging works for deep learning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI'19/IAAI'19/EAAI'19).* AAAI Press, Article 698, 8 pages.

[67] Yuhao Zhang, Frank McQuillan, Nandish Jayaram, Nikhil Kak, Ekta Khanna, Orhan Kislal, Domino Valdano, and Arun Kumar. 2021. Distributed deep learning on data systems: a comparative analysis of approaches. *Proc. VLDB Endow.* 14, 10 (jun 2021), 1769–1782.

[68] Lixi Zhou, Jiaqing Chen, Amitabh Das, Hong Min, Lei Yu, Ming Zhao, and Jia Zou. 2022. Serving deep learning models with deduplication from relational databases. *Proc. VLDB Endow.* 15, 10 (jun 2022), 2230–2243.

[69] 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.