







AdaBest: Minimizing Client Drift in Federated Learning via Adaptive Bias Estimation

Farshid Varno^{1,2} , Marzie Saghaiy¹ , Laya Rafiee Sevyeri^{2,3} , Sharut Gupta^{2,4} , Stan Matwin^{1,5} , and Mohammad Havaei² 

¹ Dalhousie University, Halifax, Canada

{f.varno,m.saghayi}@dal.ca, stan@cs.dal.ca

² Imagia Cybernetics Inc., Montreal, Canada

{laya.rafiiee,sharut.gupta,mohammad.havaei}@gmail.com

³ Concordia University, Montreal, Canada

⁴ Indian Institute of Technology Delhi, New Delhi, India

⁵ Polish Academy of Sciences, Warsaw, Poland

Abstract. In Federated Learning (FL), a number of clients or devices collaborate to train a model without sharing their data. Models are optimized locally at each client and further communicated to a central hub for aggregation. While FL is an appealing decentralized training paradigm, heterogeneity among data from different clients can cause the local optimization to *drift* away from the global objective. In order to estimate and therefore remove this drift, variance reduction techniques have been incorporated into FL optimization recently. However, these approaches inaccurately estimate the clients’ drift and ultimately fail to remove it properly. In this work, we propose an adaptive algorithm that accurately estimates drift across clients. In comparison to previous works, our approach necessitates less storage and communication bandwidth, as well as lower compute costs. Additionally, our proposed methodology induces stability by constraining the norm of estimates for client drift, making it more practical for large scale FL. Experimental findings demonstrate that the proposed algorithm converges significantly faster and achieves higher accuracy than the baselines across various FL benchmarks.

Keywords: Federated Learning, Distributed Learning, Client Drift, Biased Gradients, Variance Reduction

1 Introduction

In Federated Learning (FL), multiple sites with data often known as *clients* collaborate to train a model by communicating parameters through a central hub called *server*. At each round, the server broadcasts a set of model parameters to a number of clients. Selected clients separately optimize towards their local objective. The locally trained parameters are sent back to the server, where they are aggregated to form a new set of parameters for the next round. A well-known aggregation is to simply average the parameters received from the participating clients in each round. This method is known as FEDAVG [17] or LOCALSGD [25].

In order to reduce the communication costs as well as privacy concerns [32], multiple local optimization steps are often preferable and sometimes inevitable [17]. Unfortunately, multiple local updates subject the parameters to *client drift* [7]. While SGD is an unbiased gradient descent estimator, LOCALSGD is biased due to the existence of client drift. As a result, LOCALSGD converges to a neighborhood around the optimal solution with a distance proportionate to the magnitude of the bias [2]. The amount of this bias itself depends on the heterogeneity among the clients’ data distribution, causing LOCALSGD to perform poorly on non-iid benchmarks [31].

One effective way of reducing client drift is by adapting Reduced Variance SGD (RV-SGD) methods [6,22,23,19] to LOCALSGD. The general strategy is to regularize the local updates with an estimate of gradients of inaccessible training samples (i.e., data of other clients). In other words, the optimization direction of a client is modified using the estimated optimization direction of other clients. These complementary gradients could be found for each client i by subtracting an estimate of the local gradients from an estimate of the oracle’s⁶ full gradients. In this paper, we refer to these two estimates with \mathbf{h}_i and \mathbf{h} , respectively. Therefore, a reduced variance local gradient for client i would be in general form of $\nabla L_i + (\mathbf{h} - \mathbf{h}_i)$ where ∇L_i corresponds to the true gradients of the local objective for client i .

The majority of existing research works on adapting RV-SGD to distributed learning do not meet the requirement to be applied to FL. Some proposed algorithms require full participation of clients [24,21,15], and thus are not scalable to *cross-device* FL⁷. Another group of algorithms require communicating the true gradients [14,18] and, as a result, completely undermine the FL privacy concerns such as attacks to retrieve data from true gradients [32].

SCAFFOLD [7] is an algorithm that supports partial participation of clients and does not require the true gradients at the server. While SCAFFOLD shows superiority in performance and convergence rate compared to its baselines, it consumes twice as much bandwidth. To construct the complementary gradients, it computes and communicates \mathbf{h} as an extra set of parameters to each client along with the model parameters. FEDDYN [1] proposed to apply \mathbf{h} in a single step prior to applying any local update, and practically found better performance and convergence speed compared to SCAFFOLD. Since applying \mathbf{h} uses the same operation in all participating clients, [1] moved it to the server instead of applying on each client. This led to large savings of local computation, and more importantly to use the same amount of communication bandwidth as vanilla LOCALSGD (i.e., FEDAVG), which is half of what SCAFFOLD uses.

FEDDYN make several assumptions that are often not satisfied in large-scale FL. These assumptions include having prior knowledge about the total number of clients, a high rate of re-sampling clients, and drawing clients uniformly from

⁶ Oracle dataset refers to the hypothetical dataset formed by stacking all clients’ data. Oracle gradients are the full-batch gradients of the Oracle dataset.

⁷ In contrast to *cross-silo* FL, cross-device FL is referred to a large-scale (in terms of number of clients) setting in which clients are devices such as smart-phones.

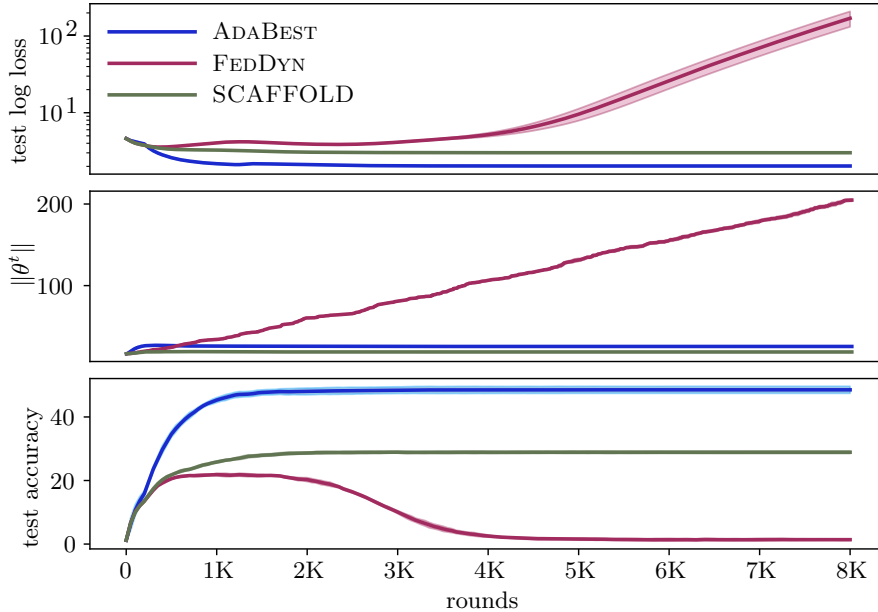


Fig. 1. Asymptotic instability of FEDDYN as a results of unbounded increase of $\|\mathbf{h}^t\|$. From top to bottom test loss (log scale), norm of cloud parameters, and test accuracy are shown in subplots. The shared horizontal axis shows the number of communication rounds. Each experiment is repeated 5 times with different random seed of data partitioning. Solid lines and shades represent mean and standard deviation respectively

a stationary set. Even with these assumptions, we show that \mathbf{h} in FEDDYN is pruned to explosion, especially in large-scale setting. This hurts the performance and holds the optimization back from converging to a stationary point.

This paper proposes ADABEST, a Reduced Variance LocalSGD (RV-LSGD) solution to minimize the client drift in FL. Compared to the baselines, we define a simpler yet more elegant method of incorporating previous observations into estimates of complementary gradients. Our solution alleviates the instability of the norm of \mathbf{h} in FEDDYN (see Figure 1 for empirical evaluation) while consuming the same order of storage and communication bandwidth, and even reducing the compute cost (see supplementary material for quantitative comparison). Unlike previous works, our algorithm provides a mechanism for adapting to changes in the distribution of client sampling and does not require prior knowledge of the entire set of clients. These characteristics of ADABEST, combined with its stability, provide a practical solution for large-scale cross-device FL. Our main contributions are as follows:

- We show that the existing RV-LSGD approaches for cross-device FL fail to efficiently converge to a stationary point. In particular, the norm of the parameters in FEDDYN is pruned to explosion (see section 3.4 for discussion).

- We formulate a novel arithmetic approach for implicit accumulation of previous observations into the estimates of oracle full gradients (\mathbf{h}).
- Using the new formulation, we present ADABEST, a novel algorithm that can be thought as a generalization of both FEDAVG and FEDDYN. We introduce a new factor β that stabilizes our algorithm through controlling the norm of \mathbf{h} . As a result, the optimization algorithm converges to a stationary point (see Sections 3.4 for detailed discussion). Unlike baselines, ADABEST does not assume that the set of training clients are stationary nor it requires a prior knowledge about its cardinality.
- We conduct various experiments under different settings of number of clients, balance among partitions, and heterogeneity. Our results indicate superior performance of ADABEST in nearly all benchmarks (up to 94% improvement in test accuracy compared to the second best candidate; almost twice better), in addition to significant improvements in stability and convergence rate.

2 Related Work

A major challenge in FL is data heterogeneity across clients where the local optima in the parameter space at each client may be far from that of the global optima. This causes a *drift* in the local parameter updates with respect to the server aggregated parameters. Recent research has shown that in such heterogeneous settings, FEDAVG is highly pruned to client drift [31].

To improve the performance of FL with heterogeneous data, some previous works use knowledge distillation to learn the cloud model from an ensemble of client models. This approach has been shown to be more effective than simple parameter averaging in reducing bias of the local gradients [16,12,33].

Another group of methods can be categorized as *gradient based* in which the gradients are explicitly constrained on the clients or sever for bias removal. FEDPROX [13] penalizes the distance between the local and cloud parameters whereas [26] normalizes the client gradients prior to aggregation. Inspired by momentum SGD, [29] uses a local buffer to accumulate gradients from previous rounds at each client and communicate the momentum buffer with the server as well as the local parameters which doubles the consumption of communication bandwidth. Instead of applying momentum on the client level, [5,27] implement a server momentum approach which avoids increasing communication costs.

Inspired by Stochastic Variance Reduction Gradients (SVRG) [6], some works incorporate variance reduction into local-SGD [1,14,7,15,30,9,18,19]. DANE [24], AIDE [21], and VRL-SGD [15] incorporated RV-SGD in distributed learning for full client participation. FEDDANE [14] is an attempt to adapt DANE to FL setting⁸, though it still undermines the privacy concerns such as attacks to retrieve data from true gradients [32]. Different from our work, most methods in this category such as VRL-SGD [15], FSVRG [9], FEDSPLIT [20] and FEDPD [30] require full participation of clients which makes them less suitable

⁸ Recall that Federated Learning is a sub-branch of distributed learning with specific characteristics geared towards practicality[17].

for cross-device setting where only a fraction of clients participate in training at each round. While FEDDANE [14] works in partial participation, empirical results show it performs worse than Federated Averaging [1]. More comparable to our method are those capable of learning in partial participation setting. In particular, SCAFFOLD [7] uses control variates on both the server and clients to reduce the variance in local updates. In addition to the model parameters, the control variates are also learned and are communicated between the server and the clients which take up additional bandwidth. [18] also reduces local variance by estimating the local bias at each client and using an SVRG-like approach to reduce the drift. While SCAFFOLD applied variance reduction on clients, FEDDYN [1] applies it partly on the server and partly on the clients. Our proposed method, is probably closer to FEDDYN than to others; however, they differ in the way gradients are estimated. See section 3.4 for detailed comparison of ADABEST with FEDDYN and SCAFFOLD.

3 Method

In this section, we present an overview of the general FL setup and further introduce our notation to formulate the problem statement. Next we detail the proposed algorithm and how to apply it. Finally, we demonstrate the efficacy of our technique by comparing it with the most closely related approaches.

3.1 Federated Learning

We assume a standard FL setup in which a central server communicates parameters to a number of clients. The goal is to find an optimal point in the parameter space that solves a particular task while clients keep their data privately on their devices during the whole learning process.

Let S^t be the set of all registered clients at round t and \mathcal{P}^t be a subset of it drawn from a distribution $P(S^\tau; \tau = t)$. The server broadcasts the *cloud model* θ^{t-1} to all the selected clients. Each client $i \in \mathcal{P}^t$, optimizes the cloud model based on its local objective and transmits the optimized *client model*, θ_i^t back to the server. The server aggregates the received parameters and prepares a new cloud model for the next round. Table 1 lists the most frequently used symbols in this paper along with their meanings. Note that the *aggregate model* (client gradients) is an average of *client model* (client gradients) over values of $i \in \mathcal{P}^t$.

3.2 Adaptive Bias Estimation

Upon receiving the client models of round t ($\{\forall i \in \mathcal{P}^t : \theta_i^t\}$) on the server, the aggregate model, θ^t is computed by averaging them out.

Definition 1. *Pseudo-gradient of a variable u at round t is $u^{t-1} - u^t$.*

Remark 1. Aggregating client models by averaging is equivalent to applying a gradient step of size 1 from the previous round’s cloud model using average of client pseudo-gradients or mathematically it is $\theta^t \leftarrow \frac{1}{|S^t|} \sum_{i \in S^t} \theta_i^t = \theta^{t-1} - \bar{g}^t$.

Table 1. Summary of notion used in this paper

| | |
|---|---|
| $u^t, u_i^t, u_i^{t,\tau}$ | variable u at {round t , and client i , and local step τ } |
| $ \cdot , \langle \cdot, \cdot \rangle, u^{(v)}$ | cardinality, inner product, power |
| $\ \cdot\ ^2, \angle(\cdot, \cdot)$ | 2-norm squared, angle |
| S^t, \mathcal{P}^t | set of {all, round} clients |
| $\theta^t, \bar{\theta}^t, \theta_i^t, \theta_i^{t,\tau}$ | {cloud, aggregate, client, local} model |
| $g^t, \bar{g}^t, g_i^t, g_i^{t,\tau}$ | {oracle, aggregate, client, local} gradients |
| h^t, h_i^t | {full, client} gradients estimates |

Next, the server finds the *cloud model* θ^t by applying the estimate of the oracle gradients h^t ; that is

$$\theta^t \leftarrow \bar{\theta}^t - h^t, \quad (1)$$

where h^t is found as follows

$$h^t = \beta(\bar{\theta}^{t-1} - \bar{\theta}^t). \quad (2)$$

In section 3.5, we further discuss the criteria for chosen β which leads to a fast convergence. The described cycle continues by sending the cloud model to the clients sampled for the next round ($t + 1$) while the aggregate model ($\bar{\theta}^t$) is retained on the server to be used in calculation of h^{t+1} or deployment. A schematic of the geometric interpretation of the additional drift removal step taken at the server is shown in Figure 2.

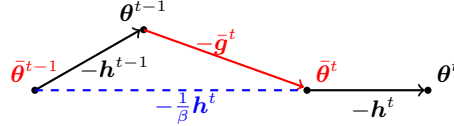


Fig. 2. Geometric interpretation of ADABEST's correction applied to the server updates. Server moves the aggregate parameters in the direction of $\bar{\theta}^{t-1} - \bar{\theta}^t$ before sending the models to the next round's clients

After receiving the cloud model, each client $i \in \mathcal{P}^t$, optimizes its own copy of the model towards its local objective, during which the drift in the local optimization steps is reduced using the client's pseudo-gradients stored from the previous rounds (see Algorithm 1). The modified client objective is $\arg \min_{\theta} \mathfrak{R}_i(\theta^t)$ where $\mathfrak{R}_i(\theta^t) = L_i(\theta^t) - \mu \langle \theta^t, h_i^{t'_i} \rangle$, where L_i is the local empirical risk defined by the task and data accessible by the client i , and t'_i is the last round that client i participated in the training. Accordingly, the local updates with step size η becomes

$$\theta_i^{t,\tau} \leftarrow \theta_i^{t,\tau-1} - \eta(\nabla L_i(\theta_i^{t,\tau-1}) - \mu h_i^{t'_i}), \quad (3)$$

where μ is the *regularization factor* (FEDDYN has a similar factor; see supplementary material for further discussion on the choice of optimal value for μ).

After the last local optimization step, each sampled client updates the estimate for its own local gradients and stores it locally to be used in the future rounds that the client participate in the training. This update is equivalent to $\mathbf{h}_i^t = \frac{1}{t-t_i'} \mathbf{h}^{t_i'} + \mathbf{g}_i^t$ where $\mathbf{g}_i^t = \boldsymbol{\theta}^{t-1} - \boldsymbol{\theta}_i^t$. Finally, the participating clients send the optimized model $\boldsymbol{\theta}_i^t$ back to the server. Our method along with SCAFFOLD and FEDDYN is presented in Algorithm 1.

3.3 Relation to RV-SGD

Stochastic Variance Reduction Gradients (SVRG) [6] and its variants [19,4,8,3,28] are of the most recent and popular RV-SGD algorithms. Given parameters \mathbf{w} and an objective function ℓ to minimize it modifies SGD update from $\mathbf{w}^k \leftarrow \mathbf{w}^{k-1} - \eta \nabla \ell(\mathbf{w}^{k-1}, \mathbf{x}_k)$ to

$$\mathbf{w}^k \leftarrow \mathbf{w}^{k-1} - \eta (\nabla \ell(\mathbf{w}^{t-1}, \mathbf{x}_k) + \mathcal{G}(\tilde{\mathbf{w}}) - \ell(\tilde{\mathbf{w}}, \mathbf{x}_k)), \quad (4)$$

where \mathbf{x}_k is a sample of data, $\tilde{\mathbf{w}}$ is a snapshot of \mathbf{w} in the past and $\mathcal{G}(\tilde{\mathbf{w}})$ is full batch gradients at $\tilde{\mathbf{w}}$. The analytic result of this unbiased modification is that if the empirical risk is strongly convex and the risk function over individual samples are both convex and L-smooth then the error in estimating gradients of $\mathcal{G}(\tilde{\mathbf{w}})$ is not only bounded but also linearly converges to zero (refer to [6] for proof). Under some conditions, [3] showed that this convergence rate is not largely impacted if a noisier estimate than original $\mathcal{G}(\tilde{\mathbf{w}})$ proposed by SVRG is chosen for full batch gradients. [4] investigated applying SVRG in non-convex problems and [8] generalized it for mini-batch SGD. [19] proposed SARA, a biased version of SVRG that progressively updates the estimate for full gradients for optimizations steps applied in between taking two snapshots. Our algorithm could be thought as a distributed variant of SARA where

1. $\mathcal{G}(\tilde{\mathbf{w}})$ is approximated by biased pseudo-gradients (and renamed to \mathbf{h}).
2. The outer loop for taking the snapshot is flattened using an exponential weighted average.

3.4 Relation to FL Baselines

Algorithm 1 demonstrates where our method differs from the baselines by color codes. Compared to the original SCAFFOLD, we made a slight modification in the way communication to the server occurs, preserving a quarter of the communication bandwidth usage. We refer to this modified version as SCAFFOLD/m. In the rest of this section, we will discuss the key similarities and differences between our algorithm, FEDDYN and SCAFFOLD in terms of cost, robustness and functionality.

Algorithm 1 SCAFFOLD/m, FEDDYN, and ADABEST

Input: T, θ^0, μ, β

for $t = 1$ **to** T **do**

Sample clients $\mathcal{P}^t \subseteq S^t$.

Transmit θ^{t-1} to each client in \mathcal{P}^t

Transmit h^{t-1} to each client in \mathcal{P}^t (SCAFFOLD/m)

for each client $i \in \mathcal{P}^t$ **in parallel do**

/* receive cloud model */

$\theta_i^{t,0} \leftarrow \theta^{t-1}$

/* locally optimize for K local steps */

for $k = 1$ **to** K **do**

Compute mini-batch gradients $L_i(\theta_i^{t,k-1})$

$g_i^{t,k-1} \leftarrow \nabla L_i(\theta_i^{t,k-1}) - h_i^{t'} + h^t$ (SCAFFOLD/m)

$g_i^{t,k-1} \leftarrow \nabla L_i(\theta_i^{t,k-1}) - h_i^{t'} - \mu(\theta^{t-1} - \theta_i^{t,k-1})$ (FEDDYN)

$g_i^{t,k-1} \leftarrow \nabla L_i(\theta_i^{t,k-1}) - h_i^{t'}$ (ADABEST)

$\theta_i^{t,k} \leftarrow \theta_i^{t,k-1} - \eta g_i^{t,k-1}$

end for

/* update local gradient estimates */

$g_i^t \leftarrow \theta^{t-1} - \theta_i^{t,K}$

$h_i^t \leftarrow \frac{|S^t|-1}{|S^t|} h_i^{t-1} + \frac{|\mathcal{P}^t|}{K\eta|S^t|} (\theta^{t-1} - \bar{\theta}^t)$ (SCAFFOLD/m)

$h_i^t \leftarrow h_i^{t'} + \mu g_i^t$ (FEDDYN)

$h_i^t \leftarrow \frac{1}{t-t'} h_i^{t'} + \mu g_i^t$ (ADABEST)

$t'_i \leftarrow t$

Transmit client model $\theta_i^t := \theta_i^{t,K}$.

end for

/* aggregate received models */

$\bar{\theta}^t \leftarrow \frac{1}{|\mathcal{P}^t|} \sum_{i \in \mathcal{P}^t} \theta_i^t$

/* update oracle gradient estimates */

$h^t \leftarrow \frac{|S^t|-1}{|S^t|} h^{t-1} + \frac{|\mathcal{P}^t|}{K\eta|S^t|} (\theta^{t-1} - \bar{\theta}^t)$ (SCAFFOLD/m)

$h^t \leftarrow h^{t-1} + \frac{|\mathcal{P}^t|}{|S^t|} (\theta^{t-1} - \bar{\theta}^t)$ (FEDDYN)

$h^t \leftarrow \beta(\bar{\theta}^{t-1} - \bar{\theta}^t)$ (ADABEST)

/* update cloud model */

$\theta^t \leftarrow \bar{\theta}^t$ (SCAFFOLD/m)

$\theta^t \leftarrow \bar{\theta}^t - h^t$ (FEDDYN)

$\theta^t \leftarrow \bar{\theta}^t - h^t$ (ADABEST)

end for

Cost SCAFFOLD consumes twice as much communication bandwidth as FEDDYN and ADABEST. This should be taken into account when comparing the experimental performance and convergence rate of these algorithms. All of these three algorithms require the same amount of storage on the server and on each client. Finally, ADABEST has a lower compute cost compared to FEDDYN, SCAFFOLD both locally (on clients) and on the server. We provide quantitative comparison of these costs in supplementary material.

Robustness According to the definition of cross-device FL, the number of devices could even exceed the number of examples per each device [17]. In such a massive pool of devices, if the participating devices are drawn randomly at uniform (which our baselines premised upon), there is a small chance for a client to be sampled multiple times in a short period of time. In FEDDYN, however, $\mathbf{h}^t = \sum_{\tau=1}^t \frac{|\mathcal{P}^\tau|}{|S^\tau|} \bar{\mathbf{g}}^\tau$, making it difficult for the norm of \mathbf{h} to decrease if pseudo-gradients in different rounds are not negatively correlated with each other (see Theorem 1 and its proof in supplementary material). In case clients are not re-sampled with a high rate then this negative correlation is unlikely to occur due to changes made to the state of the parameters and so the followup pseudo-gradients (see Section 3.5 for detailed discussion). A large norm of \mathbf{h}^t leads to a large norm of $\boldsymbol{\theta}^t$ and in turn a large $\|\bar{\boldsymbol{\theta}}^{t+1}\|^2$. This process is exacerbated during training and eventually leads to exploding norm of the parameters (see Figure 1). In Section 3.5, we intuitively justify the robustness of ADABEST for various scale and distribution of client sampling.

Theorem 1. *In FEDDYN, $\|\mathbf{h}^t\|^2 \leq \|\mathbf{h}^{t-1}\|^2$ requires*

$$\cos(\angle(\mathbf{h}^{t-1}, \bar{\mathbf{g}}^t)) \leq -\frac{|\mathcal{P}^t|}{2|S^t|} \frac{\|\bar{\mathbf{g}}^t\|}{\|\mathbf{h}^{t-1}\|}. \quad (5)$$

Functionality ADABEST allows to control how far to look back through the previous rounds for effective estimation of full and local gradients compared to existing RV-LSGD baselines. To update the local gradient estimates, we dynamically scale the previous values down because the period between computing and using $\mathbf{h}_i^{t'_i}$ on client i (the period between two participation, i.e., $t - t'_i$) can be long during which the error of estimation may notable increase. See Algorithm 1 for comparing our updates on local gradients estimation compared to that of the baselines. Furthermore, at the server, \mathbf{h}^t is calculated as the weighted difference of two consecutive aggregate models. Note that, if expanded as a power series, this difference by itself is equivalent to accumulating pseudo-gradients across previous rounds with an exponentially weighted factor. This series is presented in Remark 3 for which the proof is provided in the supplementary material. Unlike previous works, proposed pseudo-gradients' accumulation does not necessitate any additional effort to explicitly maintain information about the previous rounds. Additionally, it reduces the compute cost as quantitatively shown in the

supplementary material. It is a general arithmetic; therefore, could be adapted to work with our baselines as well.

Remark 2. $\bar{\theta}^{t-1} - \bar{\theta}^t$ is equivalent to $\mathbf{h}^{t-1} + \bar{\mathbf{g}}^t$ in ADABEST.

Remark 3. Cloud pseudo-gradients of ADABEST form a power series of $\mathbf{h}^t = \sum_{\tau=1}^t \beta^{(t-\tau)} \bar{\mathbf{g}}^\tau$, given that superscript in parenthesis means power.

3.5 Adaptability

As indicated earlier, the error in estimation of oracle full gradients in FEDDYN is only supposed to be eliminated by using pseudo-gradients. A difficult learning task, both in terms of optimization and heterogeneity results in a higher variance of pseudo-gradients when accompanies with a low rate of client participation. The outcome of constructing a naive estimator by accumulating these pseudo-gradients is sever. This is shown in Figure 1, where, on average, there is a long wait time between client re-samples due to the large number of participating clients. The results of this experiment empirically validates that $\|\theta^t\|^2$ in FEDDYN grows more rapidly and to a much higher value than ADABEST. SCAFFOLD is prone to the same type of failure; however, because it scales down previous values of \mathbf{h} in its accumulation, the outcomes are less severe than that of FEDDYN. In the supplementary material, we present, similar analysis, for a much simpler task (classification on EMNIST-L). It is important not to confuse the source of FEDDYN’s instability with overfitting (see supplementary material for overfitting analysis). However, our observations imply that the stability of FEDDYN decreases with the difficulty of the task.

Our parameter β solves the previously mentioned problem with norm of \mathbf{h} . It is a scalar values between 0 and 1 that acts as a slider knob to determine the trade-off between the amount of information maintained from the previous estimate of full gradient and the estimation that a new round provides. On an intuitive level, a smaller β is better to be chosen for a more difficult task (both in terms of optimization and heterogeneity) and lower level participation—and correspondingly higher round to round variance among pseudo-gradients and vice versa. We provide an experimental analysis of β in the supplementary material; however, in a practical engineering level, β could be dynamically adjusted based on the variance of the pseudo-gradients. The goal of this paper is rather showing the impact of using β . Therefore, we tune it like other hyper-parameters in our experiments. We leave further automation for finding an optimal β to be an open question for the future works.

Remark 4. FEDAVG is a special case of ADABEST where $\beta = \mu = 0$.

Remark 5. Server update of FEDDYN is a special case of ADABEST where $\beta = 1$ except that **an extra $\frac{|P|}{|S|}$ scalar is applied which also adversely makes FedDYN require prior knowledge about the number of clients.**

Theorem 2. *If S be a fixed set of clients, $\bar{\theta}$ does not converge to a stationary point unless $\mathbf{h} \rightarrow 0$.*

As mentioned in Section 3.4 and more particular with Theorem 1, FEDDYN is only able to decrease norm of \mathbf{h} if pseudo-gradients are negatively correlated with oracle gradient estimates which could be likely only if the rate of client re-sampling is high. Therefore, with these conditions often being not true in large-scale FL and partial-participation, it struggles to converge to an optimal point. SCAFFOLD has a weighting factor that eventually could decrease $\|\mathbf{h}\|$ but it is not controllable. Our algorithm enables a direct decay of \mathbf{h} through decaying β . We apply this decay in our experiments when norm of \mathbf{h} plateaus (see Section 4.4). This is consistent with Theorem 2 which states that converging to a stationary point require $\mathbf{h} \rightarrow 0$.

4 Experiments

4.1 Setup

We evaluate performance and speed of convergence of our algorithm against state-of-the-art baselines. We concentrate on FL classification tasks defined using three well-known datasets. These datasets are, the letters classification task of EMNIST-L [11] for an easy task, CIFAR10 [10] for a moderate task and CIFAR100 [10] for a challenging task. The training split of the dataset is partitioned randomly into a predetermined number of clients, for each task. 10% of these clients are set aside as validation clients and only used for evaluating the performance of the models during hyper-parameter tuning. The remaining 90% is dedicated to training. The entire test split of each dataset is used to evaluate and compare the performance of each model. Our assumption throughout the experiments is that, test dataset, oracle dataset, and collective data of validations clients have the same underlying distribution.

To ensure consistency with previous works, we follow [1] to control heterogeneity and sample balance among client data splits. For heterogeneity, we evaluate algorithms in three modes: IID, $\alpha = 0.3$ and $\alpha = 0.03$. The first mode corresponds to data partitions (clients' data) with equal class probabilities. For the second and third modes, we draw the skew in each client's labels from a Dirichlet distribution with a concentration parameter α . For testing against balance of sample number, we have two modes: *balanced* and *unbalanced* such that in the latter, the number of samples for each client is sampled from a log-normal distribution with concentration parameter equal to 0.3.

Throughout the experiments we consistently keep the local learning rate, number of local epochs and batch size as 0.1, 5 and 45 respectively. Local learning rate is decayed with a factor of 0.998 at each round. As tuned by [1], the local optimizer uses a weight decay of 10^{-4} for the experiments on EMNIST-L and 10^{-3} for the experiment on CIFAR10 and CIFAR100. Further details about the optimization is provided in supplementary material.

To tune the hyper-parameters we first launch each experiment for 500 rounds. μ of FEDDYN is chosen from $[0.002, 0.02, 0.2]$, with 0.02 performing best in all cases except EMNIST-L, where 0.2 also worked well. For the sake of consistency,

we kept $\mu = 0.02$ for ADABEST as well. We found the rate of client participation to be an important factor for choosing a good value for β . Therefore, for 1% client participation experiments we search β in $[0.2, 0.4, 0.6]$. For higher rates of client participation, we use the search range of $[0.94, 0.96, 0.98, 1.0]$. For all these cases, 0.96 and 0.98 are selected for 10% and 100% client participation rates, respectively (both balanced and unbalanced). We follow [1] for choosing the inference model by averaging client models through the rounds. Experiments are repeated 5 times, each with a different random seed of data partitioning. The reported mean and standard deviation of the performance are calculated based on the the last round of these 5 instance for each setting.

4.2 Model Architecture

We use the same model architectures as [17] and [1]. For EMNIST-L, the architecture comprises of two fully-connected layers, each with 100 hidden units. For CIFAR10 and CIFAR100, there are two convolutional layers with 5×5 kernel size and 64 kernels each, followed by two fully-connected layers with 394 and 192 hidden units, respectively.

4.3 Baselines

We compare the performance of ADABEST against FEDAVG [17], SCAFFOLD [7] and FEDDYN [1]. These baselines are consistent with the ones that the closest work to us [1], has experimented with⁹. However, we avoided their choice of tuning the number of local epochs since we believe it does not comply with a fair comparison in terms of computation complexity and privacy loss.

4.4 Evaluation

Table 2 compare the performance of our model to all the baselines in various settings with 100 clients. The results show that our algorithm is effective in all settings. The 1000-device experiments confirm our arguments about the large-scale cross-device setting and practicality of ADABEST in comparison to the baselines. Our algorithm has notable gain both in the speed of convergence and the generalization performance. This gain is only reduced for some benchmarks in full client participation settings (CP=100%) where the best β is chosen close to one. According to Remark 5, and the fact that in full participation $\frac{|\mathcal{P}^t|}{|\mathcal{S}^t|} = 1$ and $t_i = t'_i + 1$ for all feasible i and t , FEDDYN and ADABEST become nearly identical in these settings. In Figure 3, we show the impact of scaling the number of clients in both balanced and imbalanced settings for the same dataset and the same number of clients sampled per round (10 clients). During the hyperparameter tuning we noticed that the sensitivity of FEDDYN and ADABEST to their μ is small specially for the cases with larger number of clients.

⁹ FEDDYN additionally compares with FEDPROX [13]; however, as shown in their benchmarks it performs closer to FEDAVG than the other baselines.

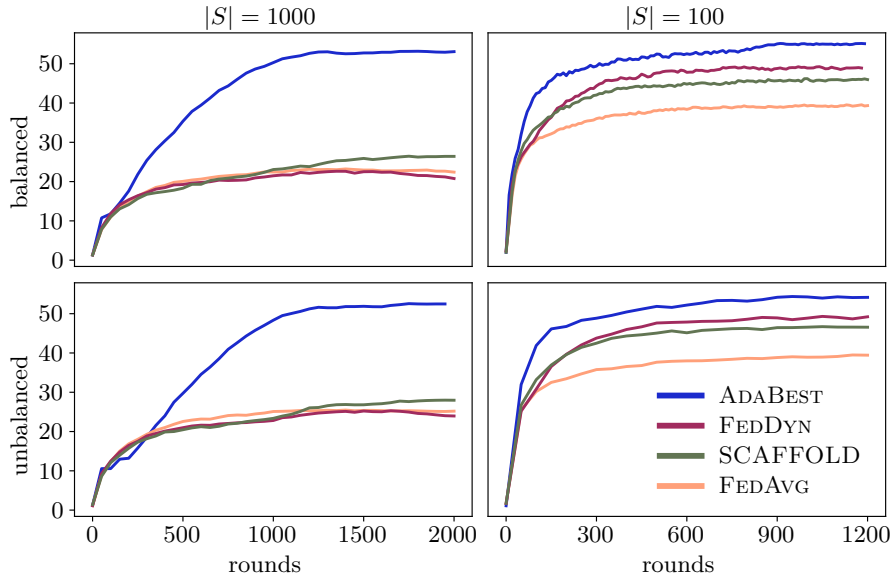


Fig. 3. Test accuracy on balanced (top) and unbalanced (bottom) settings for training on 1000 (left) and 100 (right) clients. The training dataset is CIFAR100 and $|\mathcal{P}|=10$

5 Conclusions

In this paper, we introduce ADABEST, an adaptive approach for tackling client drift in Federated Learning. Unlike the existing solutions, our approach is robust to low rates of clients re-sampling, which makes it practical for large-scale cross-device Federated Learning. Our performance and empirical convergence rates demonstrate the efficacy of our technique compared to all the baselines across various benchmarks. We have a gain of up to 0.94% in test accuracy with respect to the second best candidate which is nearly twice as good. Our algorithm consumes no more communication bandwidth or storage than the baselines, and it even has a lower compute cost. ADABEST addresses the instability of norm of gradient estimates used in FEDDYN by adapting to the most relevant information about the direction of the client drift. Furthermore, we formulated the general estimate of oracle gradients in a much elegant arithmetic that eliminates the need for the explicit, recursive form used in the previous algorithms.

Future work for this study includes deriving theoretical bounds for our proposed algorithm. Furthermore, in the current paper, the parameter β is manually tuned to account for the trade-off between the amount of information retained from the previous estimations of oracle’s gradients and the estimation provided by a new round. Developing a method for automatically tuning β is an important direction for improving the proposed algorithm.

Table 2. Mean and standard deviation of test accuracy for various settings. The results are based on 5 random data partitioning seeds. Models are trained for 1k, 1.2k, and 2k rounds, for 1%, 10% and 100% client participation settings, respectively. A *smaller* α indicates *higher* heterogeneity. *CP stands for rate of client participation ($\frac{|P|}{|S|}$)

| | | Top-1 Test Accuracy | | | | |
|------|----------|----------------------------|------------|-------------------|-------------------|-------------------|
| CP* | Dataset | Setting | FEDAVG | FEDDYN | SCAFFOLD | ADABEST |
| 1% | EMNIST-L | $\alpha=0.03$ | 94.28±0.07 | 92.42±0.14 | 93.99±0.16 | 94.49±0.07 |
| | | $\alpha=0.3$ | 94.47±0.10 | 92.64±0.31 | 94.34±0.23 | 94.72±0.22 |
| | | IID | 94.04±1.37 | 92.89±0.14 | 94.48±0.11 | 94.81±0.08 |
| | CIFAR10 | $\alpha=0.03$ | 78.18±0.80 | 77.91±0.79 | 75.83±2.36 | 78.44±1.12 |
| | | $\alpha=0.3$ | 82.21±0.36 | 82.06±0.17 | 82.96±0.42 | 83.09±0.76 |
| | | IID | 83.84±0.17 | 83.36±0.39 | 84.18±0.26 | 85.05±0.31 |
| | CIFAR100 | $\alpha=0.03$ | 47.56±0.59 | 46.27±0.65 | 47.29±0.95 | 47.91±0.83 |
| | | $\alpha=0.3$ | 49.63±0.47 | 50.53±0.36 | 52.87±0.61 | 53.62±0.23 |
| | | IID | 49.93±0.36 | 50.85±0.38 | 53.43±0.44 | 55.33±0.44 |
| 10% | EMNIST-L | $\alpha=0.03$ | 93.58±0.25 | 93.57±0.20 | 94.29±0.11 | 94.62±0.17 |
| | | $\alpha=0.3$ | 94.04±0.04 | 93.54±0.22 | 94.54±0.11 | 94.64±0.11 |
| | | IID | 94.32±0.10 | 93.60±0.35 | 94.62±0.16 | 94.70±0.24 |
| | CIFAR10 | $\alpha=0.03$ | 74.04±0.88 | 76.85±0.91 | 77.19±1.10 | 79.64±0.58 |
| | | $\alpha=0.3$ | 79.74±0.07 | 81.91±0.19 | 82.26±0.38 | 84.15±0.36 |
| | | IID | 81.35±0.23 | 83.56±0.31 | 83.50±0.15 | 85.78±0.14 |
| | CIFAR100 | $\alpha=0.03$ | 39.18±0.56 | 44.24±0.66 | 45.80±0.36 | 48.56±0.45 |
| | | $\alpha=0.3$ | 38.78±0.35 | 48.92±0.37 | 46.34±0.43 | 54.51±0.35 |
| | | IID | 37.45±0.57 | 49.60±0.24 | 44.30±0.22 | 55.58±0.14 |
| 100% | EMNIST-L | $\alpha=0.03$ | 93.36±0.15 | 94.18±0.21 | 94.38±0.20 | 94.06±0.11 |
| | | $\alpha=0.3$ | 93.99±0.19 | 94.23±0.14 | 94.53±0.16 | 94.40±0.21 |
| | | IID | 94.06±0.33 | 94.37±0.15 | 94.63±0.10 | 94.69±0.14 |
| | CIFAR10 | $\alpha=0.03$ | 72.97±1.09 | 78.24±0.77 | 77.64±0.25 | 78.07±0.71 |
| | | $\alpha=0.3$ | 79.12±0.15 | 83.19±0.18 | 82.26±0.23 | 83.20±0.25 |
| | | IID | 80.72±0.33 | 84.39±0.20 | 83.55±0.25 | 84.75±0.17 |
| | CIFAR100 | $\alpha=0.03$ | 38.24±0.63 | 46.00±0.42 | 46.51±0.50 | 46.16±0.79 |
| | | $\alpha=0.3$ | 37.03±0.35 | 50.42±0.29 | 45.48±0.38 | 50.90±0.42 |
| | | IID | 35.92±0.48 | 50.61±0.25 | 43.73±0.23 | 51.33±0.41 |

Acknowledgments: The first author wishes to express gratitude for the financial support provided by *MITACS* and *Research Nova Scotia*. In addition, the fifth author acknowledges Natural Sciences and Engineering research Council of Canada, CHIST-ERA grant CHIST-ERA-19-XAI-0 and the Polish NCN Agency NCN(grant No. 2020/02/Y/ST6/00064). We are grateful to Sai Praneeth Karimireddy, the first author of [7], for enlightening us on the proper implementation of SCAFFOLD. William Taylor-Melanson is also acknowledged for reviewing this paper and providing numerous helpful comments.

References

1. Acar, D.A.E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., Saligrama, V.: Federated learning based on dynamic regularization. In: International Conference on Learning Representations (2020)
2. Ajalloeian, A., Stich, S.U.: On the convergence of sgd with biased gradients. arXiv preprint arXiv:2008.00051 (2020)
3. Babanezhad Harikandeh, R., Ahmed, M.O., Virani, A., Schmidt, M., Konečný, J., Sallinen, S.: Stopwasting my gradients: Practical svrg. *Advances in Neural Information Processing Systems* **28** (2015)
4. Bi, J., Gunn, S.R.: A variance controlled stochastic method with biased estimation for faster non-convex optimization. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 135–150. Springer (2021)
5. Hsu, T.M.H., Qi, H., Brown, M.: Measuring the effects of non-identical data distribution for federated visual classification. arXiv preprint arXiv:1909.06335 (2019)
6. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems* **26**, 315–323 (2013)
7. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: Stochastic controlled averaging for federated learning. In: International Conference on Machine Learning. pp. 5132–5143. PMLR (2020)
8. Konečný, J., Liu, J., Richtárik, P., Takáč, M.: Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing* **10**(2), 242–255 (2015)
9. Konečný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)
10. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11), 2278–2324 (1998)
12. Li, D., Wang, J.: Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581 (2019)
13. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* **2**, 429–450 (2020)
14. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Feddane: A federated newton-type method. In: 2019 53rd Asilomar Conference on Signals, Systems, and Computers. pp. 1227–1231. IEEE (2019)
15. Liang, X., Shen, S., Liu, J., Pan, Z., Chen, E., Cheng, Y.: Variance reduced local sgd with lower communication complexity. arXiv preprint arXiv:1912.12844 (2019)
16. Lin, T., Kong, L., Stich, S.U., Jaggi, M.: Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems* **33**, 2351–2363 (2020)
17. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: *Artificial intelligence and statistics*. pp. 1273–1282. PMLR (2017)
18. Murata, T., Suzuki, T.: Bias-variance reduced local sgd for less heterogeneous federated learning. In: International Conference on Machine Learning. pp. 7872–7881. PMLR (2021)

19. Nguyen, L.M., Liu, J., Scheinberg, K., Takáč, M.: Sarah: A novel method for machine learning problems using stochastic recursive gradient. In: International Conference on Machine Learning. pp. 2613–2621. PMLR (2017)
20. Pathak, R., Wainwright, M.J.: Fedsplit: An algorithmic framework for fast federated optimization. *Advances in Neural Information Processing Systems* **33**, 7057–7066 (2020)
21. Reddi, S.J., Konečný, J., Richtárik, P., Póczós, B., Smola, A.: Aide: Fast and communication efficient distributed optimization. arXiv preprint arXiv:1608.06879 (2016)
22. Roux, N., Schmidt, M., Bach, F.: A stochastic gradient method with an exponential convergence rate for finite training sets. *Advances in neural information processing systems* **25** (2012)
23. Shalev-Shwartz, S., Zhang, T.: Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research* **14**(2) (2013)
24. Shamir, O., Srebro, N., Zhang, T.: Communication-efficient distributed optimization using an approximate newton-type method. In: International conference on machine learning. pp. 1000–1008. PMLR (2014)
25. Stich, S.U.: Local sgd converges fast and communicates little. In: International Conference on Learning Representations (2018)
26. Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V.: Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems* **33**, 7611–7623 (2020)
27. Wang, J., Tantia, V., Ballas, N., Rabbat, M.: Slowmo: Improving communication-efficient distributed sgd with slow momentum. arXiv preprint arXiv:1910.00643 (2019)
28. Xiao, L., Zhang, T.: A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization* **24**(4), 2057–2075 (2014)
29. Yu, H., Jin, R., Yang, S.: On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In: International Conference on Machine Learning. pp. 7184–7193. PMLR (2019)
30. Zhang, X., Hong, M., Dhople, S., Yin, W., Liu, Y.: Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. arXiv preprint arXiv:2005.11418 (2020)
31. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)
32. Zhu, L., Han, S.: Deep leakage from gradients. In: *Federated learning*, pp. 17–31. Springer (2020)
33. Zhu, Z., Hong, J., Zhou, J.: Data-free knowledge distillation for heterogeneous federated learning. In: International Conference on Machine Learning. pp. 12878–12889. PMLR (2021)