AdaBest: Supplementary Material

Farshid Varno[1,2], Marzie Saghayi[1], Laya Rafiee Sevyeri[2,3], Sharut Gupta[2,4], Stan Matwin[1,5], and Mohammad Havaei[2]

[1] Dalhousie University, Halifax, Canada
{f.varno,m.saghayi}@dal.ca, stan@cs.dal.ca
[2] Imagia Cybernetics Inc., Montreal, Canada
{laya.rafiee,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

A Proofs

Remark 1. Aggregating parameters by averaging is equivalent to taking a gradient step from the current parameter state in the direction of the average of the clients' pseudo-gradients, or mathematically it is $\bar{\boldsymbol{\theta}}^t \leftarrow \frac{1}{|S^t|}\sum_{i\in S^t}\boldsymbol{\theta}_i^t = \boldsymbol{\theta}^{t-1} - \bar{\boldsymbol{g}}^t$.

Proof.

$$\begin{aligned}\bar{\boldsymbol{\theta}}^t \leftarrow \frac{1}{|S^t|}\sum_{i\in S^t}\boldsymbol{\theta}_i^t &= \boldsymbol{\theta}^{t-1} - \frac{1}{|S^t|}\sum_{i\in S^t}\boldsymbol{\theta}^{t-1} - \boldsymbol{\theta}_i^t \\ &= \boldsymbol{\theta}^{t-1} - \frac{1}{|S^t|}\sum_{i\in S^t}\boldsymbol{g}_i^t \\ &= \boldsymbol{\theta}^{t-1} - \bar{\boldsymbol{g}}^t \ .\end{aligned}$$

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

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

Proof. According to Algorithm 1,

$$\boldsymbol{h}^t \leftarrow \boldsymbol{h}^{t-1} + \frac{|\mathcal{P}^t|}{|S^t|}\bar{\boldsymbol{g}}^t.$$

Applying 2-norm squared on both sides gives

$$\|\boldsymbol{h}^t\|^2 = \|\boldsymbol{h}^{t-1}\|^2 + \left(\frac{|\mathcal{P}^t|}{|S^t|}\right)^{(2)}\|\bar{\boldsymbol{g}}^t\|^2 + 2\frac{|\mathcal{P}^t|}{|S^t|}\langle \boldsymbol{h}^{t-1}, \bar{\boldsymbol{g}}^t\rangle.$$

Considering the proposition, we have

$$\therefore \ \|\boldsymbol{h}^t\|^2 \leq \|\boldsymbol{h}^{t-1}\|^2 \implies 2\langle \boldsymbol{h}^{t-1}, \bar{\boldsymbol{g}}^t \rangle \leq -\frac{|\mathcal{P}^t|}{|S^t|}\|\bar{\boldsymbol{g}}^t\|^2.$$

with dividing both sides on some positive values, we get

$$\frac{\langle \boldsymbol{h}^{t-1}, \bar{\boldsymbol{g}}^t \rangle}{\|\bar{\boldsymbol{g}}^t\|\|\boldsymbol{h}^{t-1}\|} \leq -\frac{|\mathcal{P}^t|}{2|S^t|}\frac{\|\bar{\boldsymbol{g}}^t\|}{\|\boldsymbol{h}^{t-1}\|},$$

which is equivalent to the

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

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

We first add and remove $\boldsymbol{\theta}^{t-1}$ from the first side of the equation,

$$\begin{aligned}\bar{\boldsymbol{\theta}}^{t-1} - \bar{\boldsymbol{\theta}}^t &= \bar{\boldsymbol{\theta}}^{t-1} - \bar{\boldsymbol{\theta}}^t + \boldsymbol{\theta}^{t-1} - \boldsymbol{\theta}^{t-1} \\ &= (\bar{\boldsymbol{\theta}}^{t-1} - \boldsymbol{\theta}^{t-1}) + (\boldsymbol{\theta}^{t-1} - \bar{\boldsymbol{\theta}}^t).\end{aligned}$$

Then, we replace some terms using Equation 1 and Remark 1 to get

$$\bar{\boldsymbol{\theta}}^{t-1} - \bar{\boldsymbol{\theta}}^t = \boldsymbol{h}^{t-1} + \bar{\boldsymbol{g}}^t.$$

Remark 3. Cloud pseudo-gradients of AdaBest form a power series of $\boldsymbol{h}^t = \sum_{\tau=0}^{t} \beta^{(t-\tau+1)}\bar{\boldsymbol{g}}^\tau$, given that superscript in parenthesis means power.

Proof. We make the induction hypothesis $\boldsymbol{h}^{t-1} = \sum_{\tau=0}^{t-1} \beta^{(t-\tau)}\bar{\boldsymbol{g}}^\tau$. We need to prove that $\boldsymbol{h}^t = \sum_{\tau=0}^{t} \beta^{(t-\tau+1)}\bar{\boldsymbol{g}}^\tau$. From Algorithm 1 we have

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

Additionally, using Remark 2 it could be rewritten as

$$\boldsymbol{h}^t = \beta(\boldsymbol{h}^{t-1} + \beta\bar{\boldsymbol{g}}^t).$$

Replacing the induction hypothesis changes it to

$$\begin{aligned}\boldsymbol{h}^t &= \beta(\sum_{\tau=0}^{t-1} \beta^{(t-\tau)}\bar{\boldsymbol{g}}^\tau + \beta\bar{\boldsymbol{g}}^t) \\ &= \sum_{\tau=0}^{t-1} \beta^{(t-\tau)}\bar{\boldsymbol{g}}^{\tau+1} + \beta^{(2)}\bar{\boldsymbol{g}}^t \\ &= \sum_{\tau=0}^{t} \beta^{(t-\tau+1)}\bar{\boldsymbol{g}}^\tau.\end{aligned}$$

Remark 4. FedAvg is a special case of AdaBest where $\beta = \mu = 0$.

Proof. In AdaBest, $\mu = 0$ makes $\boldsymbol{h}_i^t$ zero for all feasible i and t. The resulting local update is identical to that of FedAvg. Similarly, $\boldsymbol{h}^t$ becomes zero at all rounds if $\beta = 0$. So AdaBest would also have the same server updates as of FedAvg.

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

Proof. According to Algorithm 1, on the server side AdaBest and FedDyn are different in their update of $\boldsymbol{h}^t$. Based on Remark 2, for $\beta = 1$ AdaBest update is $\boldsymbol{h}^t \leftarrow \boldsymbol{h}^{t-1} + \bar{\boldsymbol{g}}^t$. Comparably the same update in FedDyn is $\boldsymbol{h}^t \leftarrow \boldsymbol{h}^{t-1} + \frac{|\mathcal{P}^t|}{|S^t|}\bar{\boldsymbol{g}}^t$. Involving $\|S^t\|$ in the update means assuming that prior knowledge on number of total clients is available from the beginning of the training. On the other hand, $\beta = 0$ leads to $\boldsymbol{h}^t = 0$ and consequently the update on the cloud model become $\boldsymbol{\theta}^t \leftarrow \bar{\boldsymbol{\theta}}^t$ which is identical to the server update of FedAvg.

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

Proof. With a minor abuse in our notation for the case of SCAFFOLD/m (the difference only is applying $\boldsymbol{h}$ on the clients after $\boldsymbol{\theta}$ is sent to them), we can generally state that

$$\bar{\boldsymbol{\theta}}^t \leftarrow \boldsymbol{\theta}^{t-1} - \bar{\boldsymbol{g}}^t = \bar{\boldsymbol{\theta}}^{t-1} - (\boldsymbol{h}^{t-1} + \bar{\boldsymbol{g}}^t).$$

With S being fixed, upon $t \to \infty$, and convergence of $\bar{\boldsymbol{\theta}}$, we expect that $\boldsymbol{g}^t \to \mathbf{0}$, so the optimization does not step out of the minima. In that case. we also expect $\bar{\boldsymbol{\theta}}^t \approx \bar{\boldsymbol{\theta}}^{t-1}$. On the other hand, above formula results in $\bar{\boldsymbol{\theta}}^t \approx \bar{\boldsymbol{\theta}}^{t-1} - \boldsymbol{h}^{t-1}$ which holds if $\boldsymbol{h} \to \mathbf{0}$.

B Algorithm Notation

Following [4] and [2], for the sake of simplicity and readability, we only presented Algorithm 1 for the balanced case (in terms of number of data samples per client). According to [1], SCAFFOLD and FedDyn also need the prior knowledge about the number of clients in order to properly weight their $\boldsymbol{h}^t$ accumulation in the case of unbalance data samples. These weights are used in the form of average samples per client in [1]. We eliminate such dependency in our algorithm by progressively calculating this average throughout the training. We also confirm that applying the same modification on SCAFFOLD and FedDyn does not impede their performance nor their stability (experiments on FedDyn and SCAFFOLD still are done with their original form). For the experiments, we implemented SCAFFOLD as introduced in the original paper [4]; However, for more clarity a modification of it (SCAFFOLD/m) is contrasted to other methods in Algorithm 1 in which only the model parameters are sent back to the server (compare it to Algorithm 1 in [4]). Note that this difference in presentation is irrelevant to our arguments in this paper since the more important factor for scalability in the recursion is $\frac{|S^t|-1}{|S^t|}$.

C Algorithmic Costs

In this section, we compare compute, storage and bandwidth costs of AdaBest to that of FedAvg, SCAFFOLD/m and FedDyn.

C.1 Compute Cost

Table 3 shows the notation we use for formulating the *costs* of these algorithms. Algorithm 2 is an exact repetition of Algorithm 1 except that the compute cost of operations of interest are included as comments. These costs are summed in tables Table 4 and Table 5 for the client and server sides, respectively. According to these tables, AdaBest has lower client-side and server-side compute costs than SCAFFOLD/m and FedDyn.

Table 3. Summary of notion used to formulate the algorithm costs

Notation	Meaning
n	Number of parameters of the model := $\|\boldsymbol{\theta}\|$
g	Cost of computing local mini-batch gradients
s	Cost of summing two floating point numbers
m	Cost of multiplying two floating point numbers

C.2 Storage Cost

All the three algorithms require the same amount of storage, on the clients and the server. Each client is supposed to store a set of local gradient estimates with size n (see Table 3), noted as $\boldsymbol{h}_i^t$. Likewise, each algorithm stores the same number of variables on the server so that estimates are forwarded to the next rounds. These variables are introduced with $\boldsymbol{h}^t$ in SCAFFOLD/m and FedDyn but with $\bar{\boldsymbol{\theta}}^{t-1}$ in AdaBest.

Table 4. Comparing AdaBest to FedAvg, SCAFFOLD/m, FedDyn in their compute cost of local (client side) operations. See Algorithm 2 for more detailed comparison

Algorithm	Client side compute cost
FedAvg	$K(g+ns+nm)$
SCAFFOLD/m	$K(g+ns+nm)+2Kns+2n(s+m)$
FedDyn	$K(g+ns+nm)+3Kns+Knm+n(s+m)$
AdaBest	$K(g+ns+nm)+Kns+n(s+m)$

Table 5. Comparing AdaBest to FedAvg, SCAFFOLD/m, FedDyn in their compute cost of global (server side) operations. See Algorithm 2 for more detailed comparison

Algorithm	Server side compute cost
FedAvg	$\|\mathcal{P}^t\|ns$
SCAFFOLD/m	$\|\mathcal{P}^t\|ns+2ns+2nm$
FedDyn	$\|\mathcal{P}^t\|ns+3ns+nm$
AdaBest	$\|\mathcal{P}^t\|ns+2ns+nm$

C.3 Communication Cost

AdaBest and FedDyn are not different in the way information is communicated between the server and clients; thus, they do not differ in terms of costs of communication bandwidth. That is sending n parameters from server to each selected client at each round and receiving the same amount in the other direction (from each client to the server). The original SCAFFOLD needs doubling the amount of information communicated in each direction ($2n$). However, SCAFFOLD/m reduces this overhead to 1.5 times (of that of AdaBest) by avoiding to send the extra variables from clients' to the server (1.5 n).

More accurate costs requires exact specifications of the system design. For example, the aggregation operation on the server if done in a batch (from a buffer of client delivered parameters) requires more storage but can decrease the compute cost using multi-operand adders on floating-point mantissas such as *Wallace* or *Dadda* tree adders. However, these design choices do not appear to make a difference in the ranking of the costs for algorithms compared in this paper.

For cost estimates to be more precise, system design specifications must be considered. For instance, using multi-operand adders on floating-point mantissas, such as *Wallace* or *Dadda* tree adders, can reduce the compute cost of the aggregation operation on the server if it is performed in a batch (from a buffer of client-delivered parameters) but requires more storage. However, it does not ap-

Algorithm 2 Compute cost of SCAFFOLD/m, FedDyn, AdaBest. Operations that are common among all three algorithms are grayed out. The compute cost of other operations are shown with a comment in front of each line. The variables used to represent the cost of each micro-operation are introduced in Table 3

```
**Input:** $T, \boldsymbol{\theta}^0, \mu, \beta$
**for** $t = 1$ **to** $T$ **do**
    Sample clients $\mathcal{P}^t \subseteq S^t$.
    Transmit $\boldsymbol{\theta}^{t-1}$ 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**
        $\boldsymbol{\theta}_i^{t,0} \leftarrow \boldsymbol{\theta}^{t-1}$
        **for** $k = 1$ **to** $K$ **do**
            $k = 1$ Compute mini-batch gradients $L_i(\boldsymbol{\theta}_i^{t,k-1})$ /* $g$ */
            $\boldsymbol{g}_i^{t,k-1} \leftarrow \nabla L_i(\boldsymbol{\theta}_i^{t,k-1}) - \boldsymbol{h}_i^{t'_i} + \boldsymbol{h}^t$ (SCAFFOLD/m) /* $2ns$ */
            $\boldsymbol{g}_i^{t,k-1} \leftarrow \nabla L_i(\boldsymbol{\theta}_i^{t,k-1}) - \boldsymbol{h}_i^{t'_i} - \mu(\boldsymbol{\theta}^{t-1} - \boldsymbol{\theta}_i^{t,k-1})$ (FedDyn) /* $3ns + nm$ */
            $\boldsymbol{g}_i^{t,k-1} \leftarrow \nabla L_i(\boldsymbol{\theta}_i^{t,k-1}) - \boldsymbol{h}_i^{t'_i}$ (AdaBest) /* $ns$ */
            $\boldsymbol{\theta}_i^{t,k} \leftarrow \boldsymbol{\theta}_i^{t,k-1} - \eta \boldsymbol{g}_i^{t,k-1}$ /* $ns + nm$ */
        **end for**
        $\boldsymbol{g}_i^t \leftarrow \boldsymbol{\theta}^{t-1} - \boldsymbol{\theta}_i^{t,K}$
        $\boldsymbol{h}_i^t \leftarrow \frac{|S^t|-1}{|S^t|}\boldsymbol{h}_i^{t-1} + \frac{|\mathcal{P}^t|}{K\eta|S^t|}(\boldsymbol{\theta}^{t-1} - \bar{\boldsymbol{\theta}}^t)$ (SCAFFOLD/m) /* $2ns + 2nm$ */
        $\boldsymbol{h}_i^t \leftarrow \boldsymbol{h}_i^{t'_i} + \mu \boldsymbol{g}_i^t$ (FedDyn) /* $ns + nm$ */
        $\boldsymbol{h}_i^t \leftarrow \frac{1}{t-t'_i}\boldsymbol{h}_i^{t'_i} + \mu \boldsymbol{g}_i^t$ (AdaBest) /* $ns + nm$ */
        $t'_i \leftarrow t$
        Transmit client model $\boldsymbol{\theta}_i^t := \boldsymbol{\theta}_i^{t,K}$
    **end for**
    $\bar{\boldsymbol{\theta}}^t \leftarrow \frac{1}{|\mathcal{P}^t|}\sum_{i \in \mathcal{P}^t} \boldsymbol{\theta}_i^t$ /* $|\mathcal{P}^t|ns$ */
    $\boldsymbol{h}^t \leftarrow \frac{|S^t|-1}{|S^t|}\boldsymbol{h}^{t-1} + \frac{|\mathcal{P}^t|}{K\eta|S^t|}(\boldsymbol{\theta}^{t-1} - \bar{\boldsymbol{\theta}}^t)$ (SCAFFOLD/m) /* $2ns + 2nm$ */
    $\boldsymbol{h}^t \leftarrow \boldsymbol{h}^{t-1} + \frac{|\mathcal{P}^t|}{|S^t|}(\boldsymbol{\theta}^{t-1} - \bar{\boldsymbol{\theta}}^t)$ (FedDyn) /* $2ns + nm$ */
    $\boldsymbol{h}^t \leftarrow \beta(\bar{\boldsymbol{\theta}}^{t-1} - \bar{\boldsymbol{\theta}}^t)$ (AdaBest) /* $ns + nm$ */
    $\boldsymbol{\theta}^t \leftarrow \bar{\boldsymbol{\theta}}^t$ (SCAFFOLD/m)
    $\boldsymbol{\theta}^t \leftarrow \bar{\boldsymbol{\theta}}^t - \boldsymbol{h}^t$ (FedDyn) /* $ns$ */
    $\boldsymbol{\theta}^t \leftarrow \bar{\boldsymbol{\theta}}^t - \boldsymbol{h}^t$ (AdaBest) /* $ns$ */
**end for**
```

pear that these design decisions affect the ranking of the costs for the algorithms compared in this paper.

D Experiments Details

D.1 Evaluation

There are two major differences between our evaluation and those of prior works.

1. **We do not consider number of epochs to be a hyper-parameter.** Comparing two Federated Learning (FL) algorithms with different number of local epochs, is unfair in terms of the amount of local compute costs. Additionally, it makes it difficult to justify the impact of each algorithm on preserving the privacy of clients' data. This is because the privacy cost is found to be associated with the level of random noise in the pseudo-gradients. This randomness in turn is impacted by the number of epochs (see page 8 of [3]). For an example of comparing algorithms after tuning the number of epochs, refer to the Appendix 1 of [2] where 20 and 50 local epochs are chosen respectively for FedAvg and FedDyn in order to compare their performance on MNIST dataset.
2. **We consider a hold-out set of clients for hyper-parameter tuning.** Although, an *on the fly hyper-parameter tuning* is much more appealing in FL setting, for the purpose of studying and comparing FL algorithms, it is reasonable to consider hyper-parameters are tuned prior to the main training phase. However, using the performance of the test dataset in order to search for hyper-parameters makes the generalization capability of the algorithms that use more hyper-parameters questionable. Therefore, we set aside a separate set of training clients to tune the hyper-parameters for each algorithm individually. This may make our reported results on the baselines not exactly matching that of their original papers (different size of total training samples).

In addition, we use five distinct random seeds for data partitioning to better justify our reported performance. Throughout all the experiments, SGD with a learning rate of 0.1 is used[6] with a round to round decay of 0.998. Batch size of 45 is selected for all datasets and experiments. Whenever the last batch of each epoch is less than this number, it is capped with bootstrapping from all local training examples (which can further enhances the privacy especially for imbalance settings). We follow [2] in data augmentation and transformation. Local optimization on CIFAR10 and CIFAR100 involves random horizontal flip and cropping on the training samples both with probability of 0.5. No data augmentation is applied for experiments on EMNIST-L.

D.2 Implementation

We used PyTorch to implement our FL simulator. For data partitioning and implementation of the baseline algorithms, our simulator is inspired from [1] which is publicly shared by the authors of [2]. To further validate the correctness

[6] We followed [2] in choosing this optimization algorithm and learning rate.

of SCAFFOLD implementation, we consulted the first author of [4]. Additionally, we cross-checked most of the results our simulator yielded to the ones made by [1].

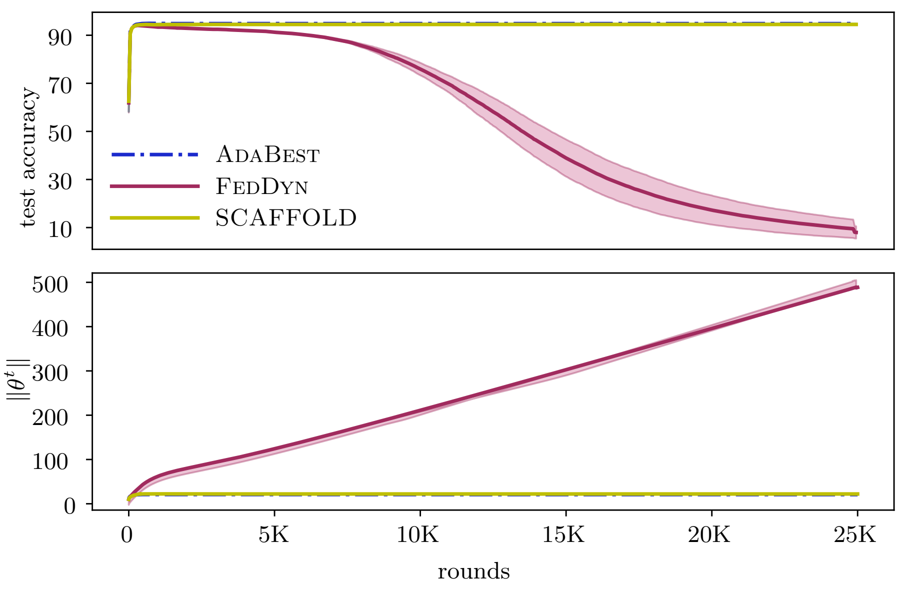

Fig. 4. Highlighting the instability of FedDyn and its association with the norm of cloud parameters. The training is performed on a comparably easy FL task but for a large number of communication rounds. Top and bottom subplots show test accuracy (in percentage) and norm of cloud parameters respectively. The horizontal axis which shows number of communication rounds is shared among subplot

D.3 Stability and $\|\boldsymbol{\theta}^t\|$

In Figure 1, we showed an experimental case of low client participation to demonstrate how $\|\boldsymbol{\theta}^t\|$ is associated with instability of FedDyn. In this experiment, the training split of CIFAR100 is partitioned over 1100 clients from which 1000 is used for training. The partitioning is balanced in terms of number of examples per client and the labels are skewed according to our $\alpha = 0.3$ heterogeneity setup (see Section 4.4 for detailed explanation). In each round, 5 clients are drawn uniformly at random. This low client participation rate is more likely to occur in a large-scale (in terms of number of clients) cross-device FL compared to the setting used for reporting the performances in Table 2 and most of those of our prior works. In Figure 4 we repeat the same experiment; except that a much simpler FL task is defined. In this experiment the training split of EMNIST-L

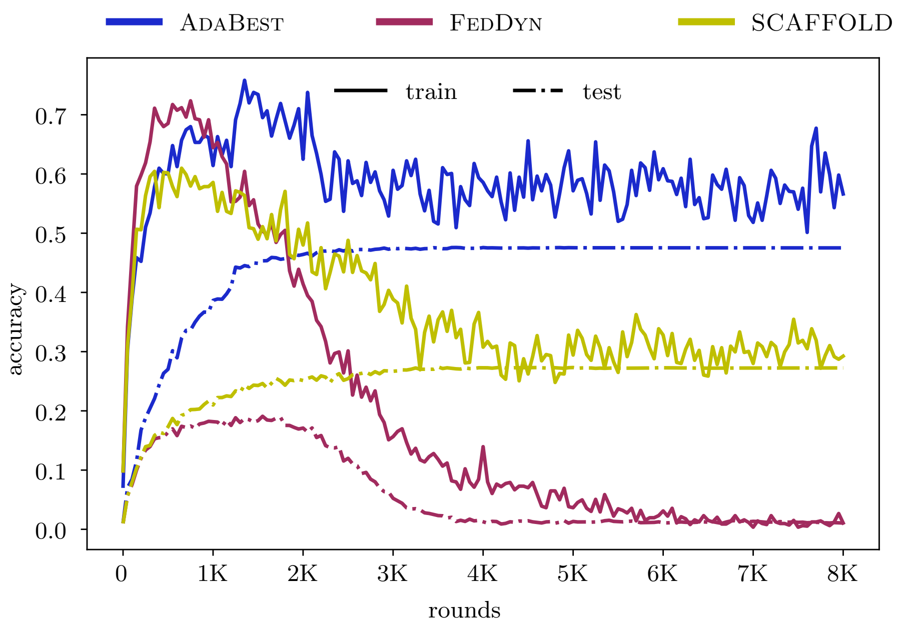

Fig. 5. Top-1 train and test accuracy score of AdaBest and the baselines. The solid and dash-dotted lines represent train and test, respectively

dataset is partitioned into 110 clients, 100 of which are used for training. Partitions are IID (labels are not skewed). Even though this task is much simpler than the previous one, still FedDyn fails to converge when the training continues for a large number of rounds.

D.4 Overfitting Analysis

It is important not to confuse FedDyn's source of instability with overfitting. To confirm this, we can compare the average train and test accuracy of the same rounds while the model is being trained. Figure 5 compares SCAFFOLD, FedDyn, and AdaBest in this manner. The configuration used in this experiment is identical to the configuration of the experiment in Figure 1, with the exception that it corresponds to a single data partitioning random seed for clarity. The train accuracy is calculated by averaging the train accuracy of participating clients. The results suggest that train accuracy of the baselines is severely declined as the training continues while AdaBest is much more stable in this regard.

D.5 μ-sensitivity

During the hyper-parameter tuning phase, we only tune β of AdaBest and set its μ constantly to 0.02. This is done throughout all the experiments except the experiment presented in this section which especially investigates the sensitivity of AdaBest to varying μ. Therefore, in practice we have not treated μ as a hyper-parameter but rather chose the value that works best for FedDyn. For this experiment, we use the same setup as the one presented in Section D.3 on EMNIST-L dataset. The only difference is that we vary μ in the range of $\{0.02 \times 2^{(k)}\}_{k=1}^{k=3}$. Figure 6 depicts the outcome of this experiment along with the result of the same analysis on FedDyn so it would be easy to compare the sensitivity of each of these algorithms to their local factor μ. As suggested by the top left subplot in this Figure, AdaBest can even achieve higher numbers in terms of the test accuracy than what is reported in Table 2. The scales on the vertical axis of the subplots related to AdaBest (on the left column) are zoomed-in to better show the stability of our algorithm throughout the training. On the other hand, the performance and stability of FedDyn shows to be heavily relied on the choice of μ when the training continues for a large number of rounds.

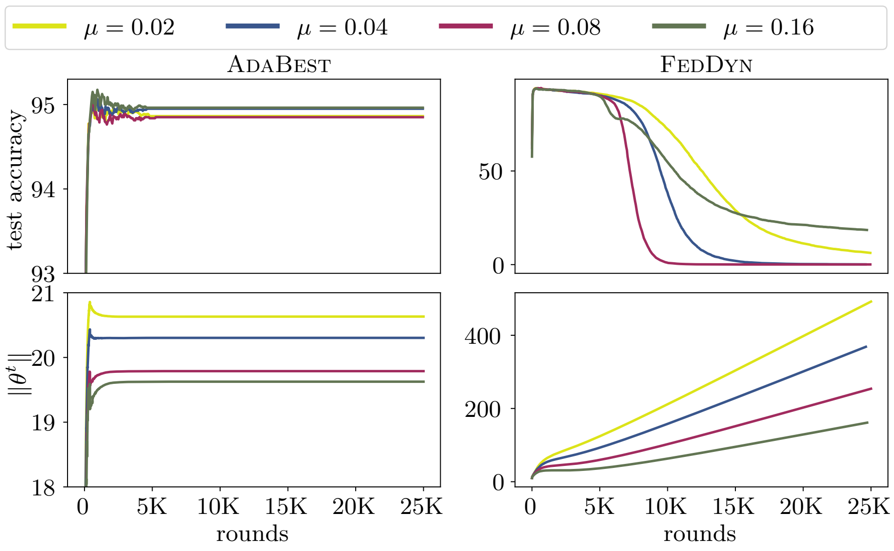

Fig. 6. μ sensitivity for AdaBest (on the left) and FedDyn (on the right). The horizontal axis which shows the number of communication rounds is shared for each column of subplots. Top and bottom row show the test accuracy (in percentage) and norm of cloud parameters respectively. Note that the vertical axis is not shared as for clarity. This means that the scale of difference between converging point of $\|\boldsymbol{\theta}^t\|$ in AdaBest is largely different from the divergence scale of the same quantity for FedDyn

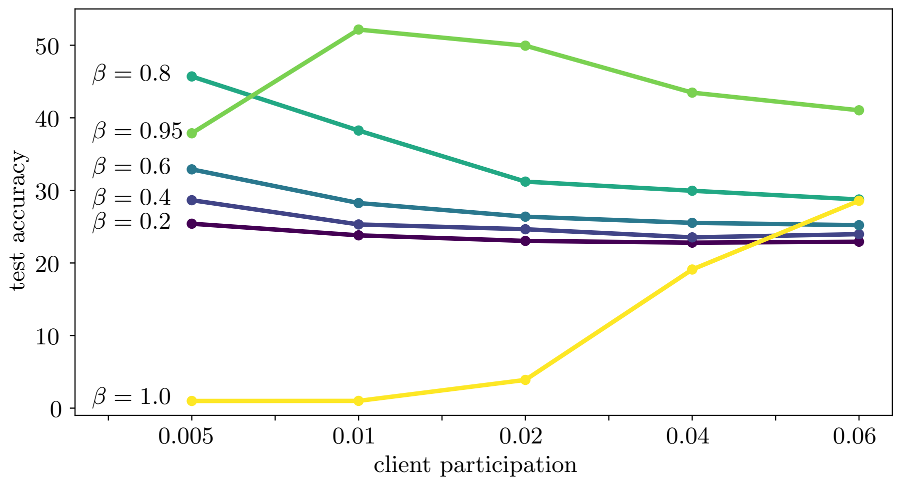

Fig. 7. The sensitivity of the test accuracy (in percentage) for different rates of client participation and values of β. The training is done on a partitioning of CIFAR100 with 1000 training clients. The numbers on the horizontal axis show the fractions of the total clients sampled at each round

D.6 $\boldsymbol{\beta}$-sensitivity

To investigate how the choice of β impacts the generalization performance, we conduct an experiment with varying β and the rate of client participation. The relation comes from the fact that in lower rates of client participation, the variance of the pseudo-gradients is higher and so a lower β is required both in order to avoid explosion of $\|\boldsymbol{\theta}^t\|$ and also to propagate estimation error from the previous rounds as explained in 3.5. For this experiment we use the same setup as the one used in Figure 1 (see Section D.3 for details). Figure 7 implies that when the rate of client participation is 0.005 (5 clients participating in each round out of 1000 training clients) the optimal β is between 0.8 and 0.95. With a larger rate of client participation, the optimal value moves away from 0.8 towards between 0.95 and 1.0;

This observation is aligned with our hypothesis on the impact of variance of pseudo-gradients on the estimation error and norm of the cloud parameters. Another interesting point is that for a wide range of β values, the performance remains almost stable regardless of the rate of client participation. Additionally, β values closer to 1 seem to be suitable for higher rates of client participation, which is suggested by the curve corresponding to $\beta = 1.0$ rising as the rate of client participation increases. This case ($\beta = 1.0$) is the most similar to formulation of our baselines where the estimations of oracle gradients are not scaled properly from one round to another.

References

1. Acar, D.A.E.: Feddyn. `https://github.com/alpemreacar/FedDyn` (2021), [Accessed 12-Mar-2022]
2. 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)
3. Fowl, L.H., Geiping, J., Czaja, W., Goldblum, M., Goldstein, T.: Robbing the fed: Directly obtaining private data in federated learning with modified models. In: International Conference on Learning Representations (2021)
4. 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)