FedADC: Accelerated Federated Learning with Drift Control

Emre Ozfatura[†], Kerem Ozfatura[‡] and Deniz Gündüz [†]
[†]Information Processing and Communications Lab, Dept. of Electrical and Electronic Engineering, Imperial College London
[‡]Department of Computer Science, Ozyegin University
{m.ozfatura,d.gunduz}@imperial.ac.uk,kerem.ozfatura@ozu.edu.tr

***Abstract*—Federated learning (FL) has become *de facto* framework for collaborative learning among edge devices with privacy concern. The core of the FL strategy is the use of stochastic gradient descent (SGD) in a distributed manner. Large scale implementation of FL brings new challenges, such as the incorporation of acceleration techniques designed for SGD into the distributed setting, and mitigation of the drift problem due to non-homogeneous distribution of local datasets. These two problems have been separately studied in the literature; whereas, in this paper, we show that it is possible to address both problems using a single strategy without any major alteration to the FL framework, or introducing additional computation and communication load. To achieve this goal, we propose FedADC, which is an accelerated FL algorithm with drift control. We empirically illustrate the advantages of FedADC.**

I. Introduction

Federated learning (FL) framework has been introduced in [1] to enable large-scale *collaborative learning* in a distributed manner and without sharing local datasets, addressing, to some extend, the privacy concerns of end-users. In FL, each participating user carries out model training using their local datasets, and exchange only their updated model parameters for consensus, with the help of a parameter server (PS). Recently, FL framework has received significant attention from both academia and industry, and has been implemented in several practical applications, such as learning keyboard prediction mechanisms on edge devices [2], [3], digital health-care /remote diagnosis [4]–[6], or for communication-efficient learning at the wireless edge [7], [8].

Large scale implementations of the FL framework introduce new challenges. One of the main obstacles in front of practical implementation of FL is that, in general, the distribution of data among end users is not homogeneous. As a result, it is possible to observe a noticeable generalization gap in practical scenarios compared to the analysis conducted under independently and identically distributed (iid) data assumption [9]–[12]. The overcome the detrimental affects of such non-iid data distribution, various modifications to the FL framework have been introduced in the recent literature [12]–[14]. In [12], it is shown that by globally sharing only a small portion of the local datasets, it is possible to tolerate the detrimental affects of non-iid distribution. However, data sharing, even of a small portion, is contradictory with the fundamental privacy-sensitive learning objective of FL, and may not be possible in certain applications. Alternatively, in [13], the authors suggest utilizing stochastic variance reduction [15] in the federated setting to control the local drifts due to non-iid data distribution. Another recent approach to control the local drift, studied in [14], is to penalize the deviation of the local model from the global one.

Another line of research explores how to employ acceleration methods in the FL framework. Acceleration methods such as momentum [16], [17], are known to be highly effective for training deep neural network (DNN) architectures, increasing both the convergence speed and the final test accuracy [18]. Recently, in [19] and [20], it has been shown that it is possible to employ different acceleration techniques at the server side to achieve better generalization error and to speed up training also in the distributed setting.

In this paper, inspired from the previous studies in [13] and [19], we introduce accelerated FL with drift control (FedADC), which uses *momentum stochastic gradient descent (SGD) optimizer* at the server for acceleration as in [19]; however, the momentum is updated through local iterations to prevent local drifts, similarly to [13]. In the SCAFFOLD strategy in [13], a globally computed gradient estimate is used to reduce the variance of the local gradient estimates.

We want to highlight that the proposed FedADC strategy does not require extra computations or additional hyper-parameters. On the other hand, it requires a momentum vector used by all the participating users. Hence, either all the participating users will track the momentum vector, even if they do not participate in a round, or the momentum needs to be sent to the participating users in each round together with the updated model, increasing the communication overhead. However, as we later discuss, this overhead can be overlapped with the computation time to prevent an increase in the communication latency. Next, we briefly provide some background on the distributed SGD strategy and how it is employed in the FL framework.

This work was funded by the European Research Council (ERC) through Starting Grant BEACON (no. 677854) and by the UK EPSRC (grant no. EP/T023600/1).

Algorithm 1 Federated Averaging (FedAvg)

```
for $t = 1, 2, \ldots$ do
    Choose a subset of users randomly $\mathcal{S}_t \subseteq [N]$: $|\mathcal{S}_t| = cN$
    for $i \in \mathcal{S}_t$ do
        Pull $\boldsymbol{\theta}_t$ from PS: $\boldsymbol{\theta}_{i,t}^{0} = \boldsymbol{\theta}_t$
        for $\tau = 1, \ldots, H$ do
            Compute SGD: $\mathbf{g}_{i,t}^{\tau} = \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}_{i,t}^{\tau-1}, \zeta_{i,\tau})$
            Update model: $\boldsymbol{\theta}_{i,t}^{\tau} = \boldsymbol{\theta}_{i,t}^{\tau-1} - \eta_t g_{i,t}^{\tau}$
        Push $\boldsymbol{\theta}_{i,t}^{H}$
    Federated Averaging: $\boldsymbol{\theta}_{t+1} = \frac{1}{|\mathcal{S}_t|} \sum_{i\in\mathcal{S}_t} \boldsymbol{\theta}_{i,t}^{H}$
```

A. Preliminaries

Consider the following collaborative learning problem across N users, each with its own local dataset denoted by $\mathcal{D}_i$, for $i = 1, \ldots, N$:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\mathbb{E}_{\zeta\sim\mathcal{D}_i} F(\boldsymbol{\theta}, \zeta)}_{:=f_i(\boldsymbol{\theta})}, \tag{1}$$

where $F(\cdot)$ is a parameterized loss function, $\boldsymbol{\theta}$ is the d-dimensional parameter model, ζ denotes a random sample, and finally $f_i(\cdot)$ is the local loss function of the i-th users.

The parallel stochastic gradient descent (PSGD) framework is designed to solve the optimization problem in (1) with the help of a PS. At the beginning of each iteration t, each user pulls the current parameter model $\boldsymbol{\theta}_t$ from the PS, and computes the *local gradient estimate*

$$\nabla_{\boldsymbol{\theta}_t} f_i(\boldsymbol{\theta}_\tau, \zeta_{i,t}), \tag{2}$$

where $\zeta_{i,t}$ is the data sampled by the i-th worker from its local dataset at iteration t. Then, each worker pushes its local gradient estimate to the PS, where those values are aggregated to update the parameter model, i.e.,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \frac{1}{N}\sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}_t} f_i(\boldsymbol{\theta}_t, \zeta_{i,t}), \tag{3}$$

where η_t is the learning rate.

In a broad sense, FL works similarly to PSGD with the following two modifications to address the communication bottleneck. In FL, instead of communicating each local gradient estimate to the PS, users carry out H local SGD iterations before sending their updated local models to the PS to obtain a consensus model. Moreover, particularly when a large number of users collaborate in FL, at each iteration, cN users are chosen randomly for the model update, where $0 \leq c \leq 1$ represent the participation ratio. The resultant FL algorithm, also known as Federated Averaging (FedAvg) is provided in Algorithm 1, where $\mathcal{S}_t$ denotes the set of chosen users at communication round t.

Algorithm 2 SLOWMO

```
for $t = 1, \ldots, T$ do
    Local iteration:
    for $i \in \mathcal{S}_t$ do in parallel
        $\boldsymbol{\theta}_{i,t}^{0} = \boldsymbol{\theta}_t$
        for $\tau = 1, \ldots, H$ do local update:
            Compute SGD: $\mathbf{g}_{i,t}^{\tau} = \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}_{i,t}^{\tau-1}, \zeta_{i,t}^{\tau})$
            Update model: $\boldsymbol{\theta}_{i,t}^{\tau} = \boldsymbol{\theta}_{i,t}^{\tau-1} - \eta_t g_{i,t}^{\tau}$
    Communication phase:
    for $i \in \mathcal{S}_t$ do
        Send $\boldsymbol{\Delta}_{i,t} = \boldsymbol{\theta}_t - \boldsymbol{\theta}_{i,t}^{H}$ to PS
    Compute pseudo gradient:
    $\bar{\mathbf{g}}_t = \frac{1}{|\mathcal{S}_t|}\frac{1}{\eta_t}\sum_{n\in\mathcal{S}_t} \boldsymbol{\Delta}_{\boldsymbol{n,t}}$
    Compute pseudo momentum:
    $\mathbf{m}_{t+1} = \beta\mathbf{m}_t + \bar{\mathbf{g}}_t$
    Model update:
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha\eta_t\mathbf{m}_{t+1}$
```

B. Background and Motivation

Recently, novel accelerated FL methods have been introduced in [19]–[21] building upon the inner/outer loop architecture [22], where the inner loop involves the local updates while the outer loop is responsible for the global update. These methods achieve better generalization error compared to the conventional FedAvg strategy. However, robustness of these solutions against the heterogeneity of data in practical FL scenarios is not discussed in the literature; although it has been demonstrated that employing an additional server-side optimizer can help to alleviate the impact of non-iid data distribution [9].

The impact of non-iid data distribution on FL has been highlighted recently by several studies [9]–[12]. The main challenge due to non-iid data distribution in FL is *local drift*, which refers to the deviations in the local models from the previous global model due to iterations based on the local dataset. These local drifts become more prominent over iterations, and lead to a higher generalization gap [9]. Although accelerated FL methods are not particularly designed to mitigate the local drift problem, we show in this paper that, with a slight alteration, server side acceleration methods can be enhanced to be robust against non-iid data distribution without an additional mechanism for drift control. Next, we explain our proposed strategy in detail.

II. Accelerated Federated Learning with Drift Control (FedADC)

The core idea behind the proposed FedADC scheme is to embed the global momentum update procedure in [19] into local iterations. In a broad sense, the local updates can be considered as a two-player game between the users and the PS, where each decides on the direction of the update alternatively. This way, one can enjoy the acceleration offered by the SLOWMO strategy in [19], while the local drifts are also controlled.

In the SLOWMO strategy [19], presented in Algorithm 2 for completeness, the inner loop, which corresponds to the local updates at the users, is identical to FedAvg in Algorithm 1. The key variation lies in the outer loop: unlike in FedAvg, where the local models are averaged to obtain the global model, the PS treats the changes in the local models as *pseudo gradients*, and utilizes them to compute a global momentum, denoted by $\mathbf{m}_t$, which is then used to update the global model. We would like to remark that the SLOWMO framework is introduced in [19] for the distributed SGD setup; and hence, the model updates are performed locally by using all-reduce communication. Here we present its adaptation to the FL setting.

Formally speaking, at the beginning of each communication round t, the PS sends the latest global model $\boldsymbol{\theta}_t$ to all the users selected to participate in the current round. Then, each selected user performs H local updates on this global model, and sends the accumulated model update $\boldsymbol{\Delta}_{n,t}$ to the PS (as illustrated in line 10 of Algorithm 2). The PS utilizes the average of the model updates as the global pseudo gradient, and updates the momentum of the outer loop accordingly (illustrated in line 14 of Algorithm 2). Finally, the PS updates the global model using the momentum just obtained (line 16 of Algorithm 2).

Remark 1. We want to remark that, in the SLOWMO framework, a second momentum can also be used for the local updates; however, in the federated setting, as the skewness of the data distribution increases, this local momentum makes the impact of the non-iid distribution even more severe.

Next, we introduce FedADC, which benefits from the SLOWMO framework with the additional robustness against non-iid data distribution. The key design trick we use here is to embed the momentum update part (illustrated in line 14 of Algorithm 3) into the local iteration; that is, instead of adding the momentum term $\mathbf{m}_t$ to the pseudo gradient at the end of the communication round, it is first normalized with respect to the number of local iterations, $\bar{\mathbf{m}}_t = \mathbf{m}_t/H$, then added to the pseudo gradient gradually through local iterations. As we have mentioned above, in Algorithm 3, local model updates are treated as a two-player game between the user and the PS, whose actions correspond to choosing the direction of the model update. The user decides on its action based on the local gradient estimate (lines 8 and 10 of Algorithm 3),while the PS decides based on the previous global pseudo momentum $\mathbf{m}_t$, specifically the normalized pseudo momentum $\bar{\mathbf{m}}_t$. By virtue of this mechanism, each worker searches for the *minima* based on its local loss function, while at the same time the PS pulls the local model towards the previous consensus direction to confine the local drift.

We consider two variations for the local updates in Algorithm 3 based on whether the actions are taken simultaneously or consecutively, illustrated with blue and red lines in Algorithm 3, respectively. One can observe that the variation illustrated with red resembles the Nesterov momentum strategy, whereas the one with blue resembles heavy ball momentum [18].

Algorithm 3 Accelerated FL with drift control (FedADC)

```
1:  for $t = 1, \dots, T$ do
2:      Local iteration:
3:      for $i \in \mathcal{S}_t$ do in parallel
4:          $\boldsymbol{\theta}_{i,t}^{0} = \boldsymbol{\theta}_t$
5:          $\bar{\mathbf{m}}_t = \mathbf{m}_t/H$
6:          for $\tau = 1, \dots, H$ do local update:
7:              $\boldsymbol{\theta}_{i,t}^{\tau-1/2} = \boldsymbol{\theta}_{i,t}^{\tau-1} - \eta_t \bar{\mathbf{m}}_t$
8:              $\mathbf{g}_{i,t}^{\tau} = \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}_{i,t}^{\tau-1/2}, \zeta_{i,t}^{\tau})$
9:              $\boldsymbol{\theta}_{i,t}^{\tau} = \boldsymbol{\theta}_{i,t}^{\tau-1/2} - \eta_t \mathbf{g}_{i,t}^{\tau}$
10:             $\mathbf{g}_{i,t}^{\tau} = \nabla_{\boldsymbol{\theta}} f_i(\boldsymbol{\theta}_{i,t}^{\tau-1}, \zeta_{i,t}^{\tau})$
11:             $\boldsymbol{\theta}_{i,t}^{\tau} = \boldsymbol{\theta}_{i,t}^{\tau-1} - \eta_t(\mathbf{g}_{i,t}^{\tau} + \bar{\mathbf{m}}_t)$
12:     Communication phase:
13:     for $i \in \mathcal{S}_t$ do
14:         Send $\boldsymbol{\Delta}_{i,t} = \boldsymbol{\theta}_t - \boldsymbol{\theta}_{i,t}^{H}$ to PS
15:     Compute Pseudo momentum:
16:     $\bar{\boldsymbol{\Delta}}_t = \frac{1}{|\mathcal{S}_t|}\frac{1}{\eta_t}\sum_{i \in \mathcal{S}_t} \boldsymbol{\Delta}_{i,t}$
17:     $\mathbf{m}_{t+1} = \bar{\boldsymbol{\Delta}}_t - (1-\beta)\mathbf{m}_t$
18:     Model update:
19:     $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha\eta_t \mathbf{m}_{t+1}$
```

We would like to stress that the overall update direction over H iterations, $\boldsymbol{\Delta}_{i,t} = \boldsymbol{\theta}_t - \boldsymbol{\theta}_{i,t}^{H}$, can be written in the following form:

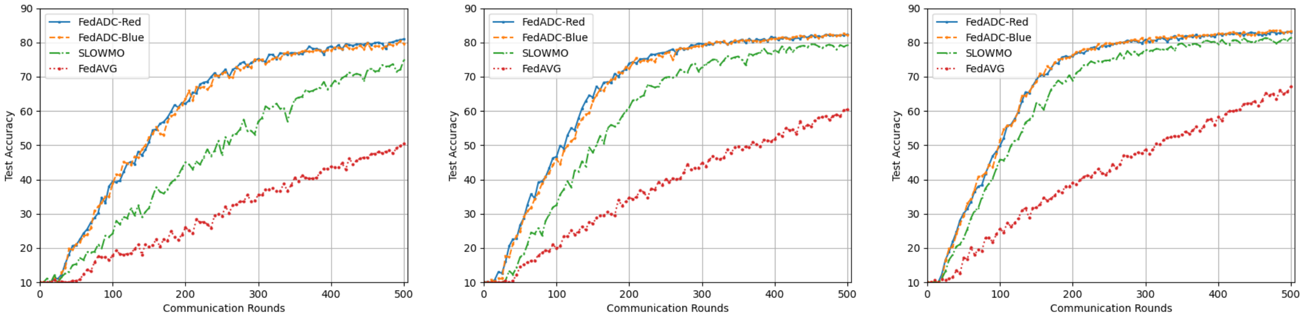
$$\boldsymbol{\Delta}_{i,t} = \eta_t \left(\sum_{\tau=0}^{H-1} \mathbf{g}_{i,t}^{\tau} + \mathbf{m}_t \right), \tag{4}$$

and the average of all the local updates is given by

$$\frac{1}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \boldsymbol{\Delta}_{i,t} = \eta_t \left(\frac{1}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \sum_{\tau=0}^{H-1} \mathbf{g}_{i,t}^{\tau} + \mathbf{m}_t \right). \tag{5}$$

If both sides of Equation (5) are divided by η_t, the right hand side of the equation is in the form $\mathbf{m}_t + \bar{\mathbf{g}}_t$. Finally, one can observe that with a small correction; that is, by subtracting $(1-\beta)\mathbf{m}_t$, the definition in (5) becomes identical to the pseudo momentum of the SLOWMO framework. Therefore, although the mechanism for the local updates is modified, the structure of the outer loop still closely resembles that of the SLOWMO strategy.

We note that, in Algorithm 3, a discounting mechanism for the momentum term is applied after the local iterations (line 16). Alternatively, discounting of the momentum can be done before the local iterations by simply setting $\bar{\mathbf{m}}_t = \beta\mathbf{m}_t/H$ in line 5 of Algorithm 3. In the most generic form, the embedding of the global momentum to local updates can be controlled by a parameter γ, such that $\bar{\mathbf{m}}_t = \gamma\beta\mathbf{m}_t/H$, and $\mathbf{m}_t$ is updated accordingly, i.e., $\mathbf{m}_{t+1} = \bar{\boldsymbol{\Delta}}_t + (1-\gamma)\beta\mathbf{m}_t$. One can easily observe that Algorithm 3 corresponds to taking $\gamma = 1/\beta$ and aforementioned alternative approach corresponds to taking $\gamma = 1$. Although different strategies can be obtained by playing with the γ parameter, we do not pursue this direction in the scope of this paper since it increases the number of hyper-parameters to be tuned.



(a) $s = 2$.

(b) $s = 3$.

(c) $s = 4$.

Fig. 1: Comparison of the convergence of test accuracy among FedADC, FedAvg and SLOWMO for the image classification task on CIFAR-10 dataset using a four-Layer CNN. We consider a different level of non-iid data distribution in each figure, parameterized by s.

Overall, the proposed FedADC strategy uses $\mathbf{m}_t$ for drift control in the inner loop and for acceleration in the outer loop. We further argue that, although the use of $\mathbf{m}_t$ in the inner loop is not similar to the momentum optimizer framework, it may still serve the common purpose of helping to escape from saddle points. In [23], it has been argued, backed by some theoretical analysis under certain assumptions (which can be validated through experiments) that the efficiency of the momentum approach is due to perturbation of the model parameters towards the escape direction from a saddle point. Hence, we argue that the fixed perturbation $\bar{\mathbf{m}}_t$, which does not depend on the local gradient estimate, may also help to escape from a saddle point. Around a saddle point, the local gradient estimates start to vanish; however, since the term $\bar{\mathbf{m}}_t$ does not depend on the local gradient estimates, vanishing behavior will not be observed for the $\bar{\mathbf{m}}_t$ term; and thus, the perturbation due to $\bar{\mathbf{m}}_t$ will prevent the local model from being stuck at a saddle point.

A. Communication load

One of the main design goals behind the FedADC strategy is to keep the number of hyper-parameters and the number of exchanged model parameters as low as possible while accelerating the training and achieving a certain robustness against non-iid data distribution and acceleration. From the uplink perspective, proposed strategy does not impose any additional communication load compared to conventional FedAvg strategy; however, on the downlink, the model difference $\bar{\mathbf{\Delta}}_t$ should be broadcast to all the users in the system, not only to those participating in the current iteration, to ensure that each user can track the global momentum $\mathbf{m}_t$.

Alternatively, at each global update phase, selected users in $\mathcal{S}_t$ can pull both the global momentum $\mathbf{m}_{t+1}$ and the global model $\boldsymbol{\theta}_t$ form the PS, which doubles the communication load in the downlink direction. However, in certain settings the additional communication load can be hidden by overlapping it with the computation time, which would prevent an increase in the overall communication latency. To clarify, at time slot t, in parallel to the computation process of users in $\mathcal{S}_t$, the PS can decide the next set of users, $\mathcal{S}_{t+1}$, and send them the current momentum $\mathbf{m}_t$ and the model $\boldsymbol{\theta}_t$; hence, at the beginning of iteration $t+1$, the users in $\mathcal{S}_{t+1}$ only need to pull the average model difference $\bar{\mathbf{\Delta}}_t$ at the beginning of iteration $t+1$. Hence, although the additional communication load at the downlink side cannot be prevented, the corresponding latency can be reduced by overlapping it with the computation time.

III. Numerical results

A. Simulation setup

For the experiments, we use the CIFAR-10 [24] image classification dataset, which contains 50,000 training and 10,000 test images from 10 classes. Training images are distributed equally among 100 users. We consider a neural network architecture with four convolutional layers and four fully connected layers with no batch normalization applied to the outputs of the layers. Max-pooling is also utilized for scaling down the image size. We set the weight decay to 4×10^{-4} in all the experiments.

B. Simulation results

To analyze the performance of the proposed FedADC scheme under non-iid data distribution, we consider the *sort and partition* approach to distribute the training dataset among the users. In the sort and partition approach, the images in the training dataset are initially sorted based on their labels, and then they are divided into blocks and distributed among the users randomly based on a parameter s, which measures the skewness of the data distribution. To be more precise, s defines the maximum number of different labels within the local dataset of each user, and therefore, the smaller s is, the more skewed the data distribution is. In our numerical experiments, we consider three different scenarios for the skewness of the data distribution with $s = 2, 3, 4$, respectively.

In all the experiments, we train the given DNN architecture for 500 communication rounds, each of which consists of $H = 8$ local iterations. We fix the user participation ratio to $c =$

0.2; that is, at each iteration only 20 users participate in the training. We use a batch size of 64.

We consider FedAvg and SLOWMO frameworks as benchmark strategies. To provide a fair comparison, we tune all the hyper-parameters including the learning rate η and the momentum coefficient β for each framework separately. For this we carry out a grid search over the values of $\eta \in \{0.01, 0.025, 0.05, 0.1\}$ and $\beta \in \{0.6, 0.7, 0.8, 0.9\}$. We fix $\alpha = 1$ similarly to [19]. For FedADC, we implement both variations illustrated with blue and red in Algorithm 3, denoted as FedADC-Blue and FedADC-Red, respectively, in the figures. Finally, we remark that each experiment is repeated 10 times, and the average test accuracy results are presented.

The convergence results of the FedADC, FedAvg and SLOWMO schemes under non-iid data distribution for $s = 2, 3, 4$ are illustrated in Fig.s 1a-1c, respectively. One can clearly observe that SLOWMO provides a significant improvement over FedAvg, but in all three scenarios, the proposed FedADC scheme outperforms SLOWMO. The improvement of SLOWMO with respect to FedAvg is consistent with the observations in [9] and [19]; that is, the use of *server side momentum* can help to mitigate the impact of local drifts, and it also accelerates learning. However, by comparing the gap between the proposed FedADC framework and SLOWMO for different s values, we also observe that SLOWMO mainly serves for acceleration rather than a drift control mechanism, thus the performance gap between FedADC and SLOWMO widens as the parameter s decreases, i.e., as the data distribution becomes more skewed.

Finally, in Fig. 2, we compare the test accuracy of the FedADC scheme for different s parameters to investigate its robustness against non-iid data distribution. We observe that, in all the cases FedADC passes %80 test accuracy within 500 communication rounds. Besides, the simulation results indicate that, although the convergence speed slows down as s decreases, it seems FedADC still converges to a similar test accuracy level for different s values, which shows the robustness of the FedADC scheme to non-iid data distribution. We also observe that when s is larger, the two local update mechanisms illustrated with red and blue in Algorithm 3 perform almost identical. However, as s decreases the Nesterov-type model updates, illustrated with red, performs slightly better as one can observe in Fig. 2.

IV. Conclusion

In this paper, we introduced a novel FL framework that is more robust to data heterogeneity across users. The proposed strategy embeds the momentum update step typically used at the server side into the local model update procedure to control the local drift and to prevent divergence. Through experiments on a CNN architecture for image classification on the CIFAR-10 dataset, we show that the proposed FedADC approach accelerates the training while also preventing local drifts, and as a result, outperforms both of the benchmarks, FedAvg and SLOWMO, in terms of the convergence speed and final test accuracy.

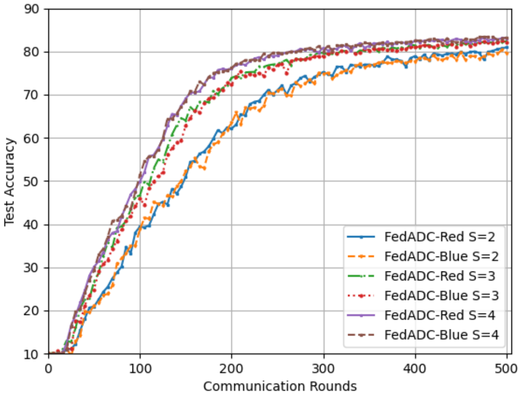


Fig. 2: Convergence performance of FedADC for non-iid data distribution with $s = 2, 3, 4$.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54. Fort Lauderdale, FL, USA: PMLR, Apr 2017, pp. 1273–1282.

[2] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR*, vol. abs/1811.03604, 2018.

[3] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *CoRR*, vol. abs/1906.04329, 2019.

[4] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng, "Privacy-preserving federated brain tumour segmentation," in *Machine Learning in Medical Imaging*. Cham: Springer International Publishing, 2019, pp. 133–141.

[5] N. Rieke, J. Hancox, W. Li, F. Milletari, H. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *CoRR*, vol. abs/2003.08119, 2020.

[6] M. Malekzadeh, B. Hasircioglu, N. Mital, K. Katarya, M. E. Ozfatura, and D. Gunduz, "Dopamine: Differentially private secure federated learning on medical data," in *Proceedings of the Second AAAI Workshop on Privacy-Preserving Artificial Intelligence (PPAI-21)*, Virtual Worskhop, Feb 2021.

[7] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *CoRR*, vol. abs/1812.02858, 2018. [Online]. Available: http://arxiv.org/abs/1812.02858

[8] D. Gunduz, D. B. Kurka, M. Jankowski, M. M. Amiri, E. Ozfatura, and S. Sreekumar, "Communicate to learn at the edge," *IEEE Communications Magazine*, Dec. 2020.

[9] T. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *CoRR*, vol. abs/1909.06335, 2019.

[10] T.-M. H. Hsu, H. Qi, and M. Brown, "Federated visual classification with real-world data distribution," *CoRR*, vol. abs/2003.08082, 2020.

[11] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. Virtual: PMLR, 13–18 Jul 2020, pp. 4387–4398.

[12] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, vol. abs/1806.00582, 2018.

[13] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the 37th International Conference on*

Machine Learning, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. Virtual: PMLR, 13–18 Jul 2020, pp. 5132–5143.

[14] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 429–450.

[15] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 314–323. [Online]. Available: http://proceedings.mlr.press/v48/reddi16.html

[16] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $o(1/k^2)$," *Proceedings of the USSR Academy of Sciences*, vol. 269, pp. 543–547, 1983.

[17] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1 – 17, 1964.

[18] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," ser. Proceedings of Machine Learning Research, vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1139–1147.

[19] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "SLOWMO: Improving communication-efficient distributed SGD with slow momentum," in *International Conference on Learning Representations*, 2020.

[20] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *CoRR*, vol. abs/2003.00295, 2020.

[21] K. Chen, H. Ding, and Q. Huo, "Parallelizing adam optimizer with blockwise model-update filtering," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 3027–3031.

[22] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton, "Lookahead optimizer: k steps forward, 1 step back," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 9597–9608.

[23] J.-K. Wang, C.-H. Lin, and J. Abernethy, "Escaping saddle points faster with stochastic momentum," in *International Conference on Learning Representations*, 2020.

[24] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (Canadian Institute for Advanced Research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html