# Federated Learning with Unbiased Gradient Aggregation and Controllable Meta Updating

**Xin Yao**[1, 2], **Tianchi Huang**[1], **Rui-Xiao Zhang**[1], **Ruiyu Li**[2], **Lifeng Sun**[*1]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2] Youtu X-Lab, Tencent, Shenzhen, China
{yaox16,htc19,zhangrx17}@mails.tsinghua.edu.cn
royryli@tencent.com, sunlf@tsinghua.edu.cn

## Abstract

Federated Averaging (FedAvg) serves as the fundamental framework in Federated Learning (FL) settings. However, we argue that 1) the multiple steps of local updating will result in gradient biases and 2) there is an inconsistency between the target distribution and the optimization objectives following the training paradigm in FedAvg. To tackle these problems, we first propose an unbiased gradient aggregation algorithm with the keep-trace gradient descent and gradient evaluation strategy. Then we introduce a meta updating procedure with a controllable meta training set to provide a clear and consistent optimization objective. Experimental results demonstrate that the proposed methods outperform compared ones with various network architectures in both the IID and non-IID FL settings.

## 1 Introduction

Federated learning (FL) [5, 6, 7] proposes leveraging the massive decentralized computing resources to perform on-device training with the local data. Federated averaging (FedAvg) [7], serving as the fundamental framework under the FL settings, selects a part of clients for participating in training in each round, then performs several epochs of local updating and finally aggregates the local models or updates on the server to get the global model. With this training paradigm, FedAvg protects the privacy of personal data and avoids heavy communication costs.

In this paper, we take a closer look at the FedAvg algorithm through theoretical analysis and argue that 1) the multiple steps of updates on clients will bring gradient biases to model aggregation (Section 2.1) and 2) selecting a part of clients for participating in training in each round will result in an inconsistency between the optimization objectives and the real targeted distribution (Section 2.2).

To tackle these two problems, we first develop an unbiased gradient aggregation algorithm (Section 3.1) with the keep-trace gradient descent and gradient evaluation strategy. Then we further introduce an additional meta updating procedure (Section 3.2) with a controllable meta training set on the central server after model aggregation in each round. Both the two improvements are model- and task-agnostic and can be applied individually or together. We conduct experiments with various network architectures, including the convolutional neural networks (CNNs) and the gated recurrent unit (GRU) network, in both the IID and non-IID FL settings. Experiments show the proposed methods are faster in convergence and achieve higher accuracy than the baselines.

To summarize, our contributions are as follows:

- We develop an unbiased gradient aggregation algorithm for FL with the keep-trace gradient descent and gradient evaluate strategy, which is compatible with the existing FedAvg framework.

This manuscript has been accepted to the Workshop on Federated Learning for Data Privacy and Confidentiality (FL - NeurIPS 2019, in Conjunction with NeurIPS 2019)

- We introduce an additional meta updating procedure on the central server. It establishes a clear and consistent objective and guides the optimization of federated models in a controllable manner.

## 2 Problem Analysis

The basic idea of FedAvg is derived from the distributed learning system consisting of parameter servers and computational workers. Concretely, let us consider the learning system containing one single parameter server and $K$ computational workers. At the beginning of round $t$, the parameter server distributes the model parameters $\omega_t$ to the workers. Then each worker $k \in K$ computes one-step gradient $g_t^{k(1)} = \nabla_{\omega_t}\mathcal{L}_k(\omega_t; \mathcal{D}_k) = \nabla_{\omega_t}\sum_{(x_i,y_i)\in\mathcal{D}_k}\ell(\omega_t; x_i, y_i)$, where $\mathcal{D}_k$ is the data distribution on client $k$ with $n_k = |\mathcal{D}_k|$ and $\ell$ is the loss function. Next, the parameter server gathers all the gradients and applies the update with weighted averaging:

$$\omega_{t+1} \leftarrow \omega_t - \eta \sum_{k\in K} \frac{n_k}{n} g_t^{k(1)} \tag{1}$$

where $n = \sum_{k\in K} n_k$ and $\eta$ is the learning rate. However, following the above updating paradigm, each of the $K$ workers has to communicate with the parameter server twice (distributing model parameter and gathering gradients) in each round, which is a heavy burden in FL settings.

For this problem, FedAvg introduces mainly two improvements. Firstly, they think Equation (1) is an equivalent to the weighted average of local parameters $\omega_{t+1}^k$:

$$\omega_{t+1} \leftarrow \omega_t - \eta \sum_{k\in K} \frac{n_k}{n} g_t^{k(1)} = \sum_{k\in K} \frac{n_k}{n}(\omega_t - \eta g_t^{k(1)}) = \sum_{k\in K} \frac{n_k}{n}\omega_{t+1}^k \tag{2}$$

Thus they add more computation to each client by iterating the local update $\omega_t^{k(i)} = \omega_t^{k(i-1)} - \eta g_t^{k(i)}$ ($i$-th step, $\omega_t^{k(0)} = \omega_t$) multiple times before sending the parameters to server.

### 2.1 Gradient Bias

Nevertheless, the weighted average of $g_t^{k(1)}$ in Equation (2) makes sense because every $g_t^{k(1)}$ is the derivative of $\omega_t$:

$$g_t^{(1)} = \nabla_{\omega_t}\mathcal{L}(\omega_t; \mathcal{D}) = \sum_{k\in K} \frac{n_k}{n}\nabla_{\omega_t}\mathcal{L}_k(\omega_t; \mathcal{D}_k) = \sum_{k\in K} \frac{n_k}{n} g_t^{k(1)} \tag{3}$$

where $\mathcal{D} = \sum_{k\in K} \frac{n_k}{n}\mathcal{D}_k$ is the overall data distribution.

We argue that Equation (3) *do not hold with multiple steps of gradient descent before averaging*. For the sake of an intuitive explanation, we take a look at the second step of gradient descent:

$$g_t^{(2)} = \nabla_{\omega_t^{(1)}}\mathcal{L}(\omega_t^{(1)}; \mathcal{D}), \qquad g_t^{k(2)} = \nabla_{\omega_t^{k(1)}}\mathcal{L}(\omega_t^{k(1)}; \mathcal{D}_k) \tag{4}$$

where $\omega_t^{(1)}$ and $\omega_t^{k(1)}$ denote the centrally and locally updated version of $\omega_t$ respectively after one-step gradient descent. Obviously, Equation (3) do not hold here because $g_t^{(2)}$ and $g_t^{k(2)}$ are the derivatives of different parameters, i.e., $\omega_t^{(1)}$ and $\omega_t^{k(1)}$ respectively. We denote the gap between $g_t$ and $\sum_{k\in K} \frac{n_k}{n} g_t^k$ as the *gradient bias*, which is rather small at the beginning but accumulates as the local updating step increases and finally harms the performance of federated models, especially in non-IID conditions [10].

### 2.2 Inconsistent Optimization Objectives

The second major improvement proposed by FedAvg is selecting a part of workers (or clients in FL settings) for performing computation in each round, i.e., replacing $K$ with $C \cdot K$, where $C$ is the fraction of clients, which actually brings another problem, i.e., *the inconsistency between the optimization objectives and the real target distribution* [8].

On round $t$, FedAvg trains the global model to minimize the empirical loss on the distribution $\mathcal{D}_{S_t}$:

$$\mathcal{D}_{S_t} = \sum_{k\in(C\cdot K)} \frac{n_k}{n_{S_t}}\mathcal{D}_k, \qquad n_{S_t} = |\mathcal{D}_{S_t}| \tag{5}$$

$S_t$ is the random set of $C \cdot K$ clients in round $t$. The aggregated gradient $g_t$ in Equation (3) indicates the descent direction on $\mathcal{D}_{S_t}$ instead of the expected $\mathcal{D}$.

There are two main reasons for this. On the one hand, $\mathcal{D}_{S_t}$ varies between rounds, which results in the lack of a clear and consistent optimization objective; On the other hand, there is a gap between $\mathcal{D}_{S_t}$ and the real target distribution $\mathcal{D}$, which may be caused by the biased selection of clients.

# 3 Method

## 3.1 Unbiased Gradient Aggregation

Taking an on-device training procedure with $E$ local epochs for example, we perform the *keep-trace gradient descent optimization* for the first $E-1$ epochs, and then evaluate gradients using the whole data in the last epoch. Then we are able to perform the Unbiased Gradient Aggregation (**UGA**).

**Keep-trace Gradient Descent:**

In the $i$-th step of updating on client $k$ in round $t$, vanilla gradient descent will execute

$$\omega_t^{k(i)} = \omega_t^{k(i-1)} - \eta g_t^{k(i)} \tag{6}$$

and keep just $\omega_t^{k(i)}$ as the initial state for the next step of updating. Denoting $\mathcal{B}_t^{k(i)}$ as the batch of examples in the $i$-th step on client $k$ in round $t$, we have:

$$g_t^{k(i)} = \nabla_{\omega_t^{k(i-1)}} \mathcal{L}(\omega_t^{k(i-1)}; \mathcal{B}_t^{k(i)}) \tag{7}$$

Notice that $g_t^{k(i)}$ is a function of $\omega_t^{k(i-1)}$ and thus $\omega_t^{k(i)}$ is also a function of $\omega_t^{k(i-1)}$, i.e., $\omega_t^{k(i)} = f_{k(i)}(\omega_t^{k(i-1)})$. Instead of treating $g_t^{k(i)}$ as numerical values and Equation (6) as a numerical computation, we *keep the functional relation* between $\omega_t^{k(i)}$ and $\omega_t^{k(i-1)}$ when updating parameters, which is termed as the keep-trace gradient descent.

**Gradient Evaluation:**

After the $E-1$ epochs of local updating, we will finally get $\omega_t^k = h_k(\omega_t)$ according to the recursive relations. Then in the last epoch, we evaluate $\omega_t^k$ on the whole client data and calculate the gradient against $\omega_t$ directly:

$$g_t^k = \nabla_{\omega_t} \mathcal{L}(\omega_t^k; \mathcal{D}_k) \tag{8}$$

Since all the $g_t^k$ for $k \in S_t$ is the derivatives of $\omega_t$, we can aggregate the gradients on the parameter server in an unbiased way using:

$$\omega_{t+1} \leftarrow \omega_t - \eta_g \sum_{k \in S_t} \frac{n_k}{n_{S_t}} g_t^k \tag{9}$$

where $\eta_g$ is the learning rate for gradient aggregation.

## 3.2 Controllable Meta Updating

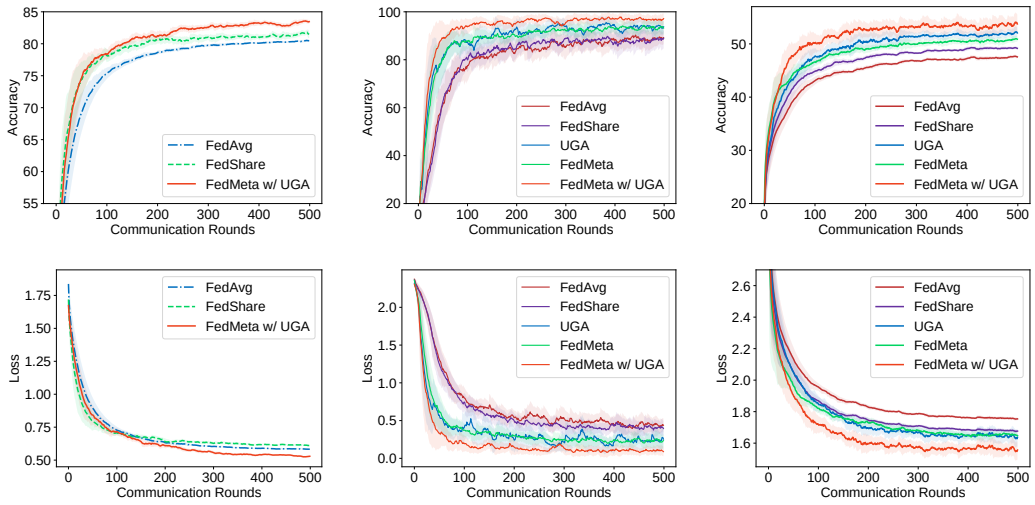
To tackle the lack of a clear and consistent objective, we introduce an additional meta updating procedure with a small set of data samples $\mathcal{D}_{meta}$ on the central server after model aggregation in each round, which is denoted as **FedMeta**.

The whole optimization process is a *two-stage optimization* that contains: 1) the inner loop optimization on clients; and 2) the outer loop optimization on the server, i.e., the meta updating procedure, using:

$$\omega_{t+1}^{meta} = \omega_{t+1} - \eta_{meta} \nabla_{\omega_{t+1}} \mathcal{L}(\omega_{t+1}; \mathcal{D}_{meta}) \tag{10}$$

where $\eta_{meta}$ is the meta learning rate. It is worth noting that in this two-stage optimization, the whole training process has a clear and consistent objective, i.e., the performance on the meta training set $\mathcal{D}_{meta}$, which solves the aforementioned problems in Section 2.2 but inevitably depends too heavily on the selection of $\mathcal{D}_{meta}$.

**The Role of $\mathcal{D}_{meta}$ and Privacy Concerns:**




(a) Split CIFAR-10 (IID) (b) FEMNIST (non-IID) (c) Shakespeare (non-IID)

Figure 1: Test accuracy (upper rwo) and loss (lower row) over communication rounds of all the methods in different FL settings. (Better viewed in color)

In a common situation, $\mathcal{D}_{meta}$ could be an IID subset of the overall data distribution $D$. In practice, it could be acquired by the data voluntarily shared by some users, or by recruiting some users for participating in the insider program and testing the beta versions of the federated applications. These methods for constructing $\mathcal{D}_{meta}$ do not violate the privacy protection principles and have been adopted in some previous studies [4, 10]. In particular, how to construct an appropriate $D_{meta}$ is not the focus of this paper.

**Controllable Federated Models:**

Further, from another point of view, $\mathcal{D}_{meta}$ *offers a way to pertinently control the behavior of federated models*. In the vanilla FedAvg, the server (or the developers behind) cannot control what kind of model is finally trained by the system. By contrast, the federated model is always optimized towards a better performance on the meta training set $\mathcal{D}_{meta}$ in FedMeta. In other words, what kind of $\mathcal{D}_{meta}$ you offer, the corresponding federated model is trained.

In fact, there is no necessary connection between the meta training set $\mathcal{D}_{meta}$ and the overall data distribution $\mathcal{D}$. Instead, $\mathcal{D}_{meta}$ should be chosen according to the specific targets, which is a powerful tool for reducing the biases and unfairness of federated models. For example, the overall data distribution $\mathcal{D}$ may contain some prejudices of gender, race or wealth [1, 3, 9, 11], but we could build a better meta training set $\mathcal{D}_{meta}$ to guide the optimization of federated models towards an unbiased and fair manner.

Pseudo-code of UGA and FedMeta are available in Appendix A.

## 4 Experiments

We evaluate the proposed methods with various network architectures and in both the IID and non-IID FL settings, i.e., CNN Model on Split CIFAR-10 (IID), CNN Model on FEMNIST (non-IID) and GRU Model on Shakespeare (non-IID)[1]. Specially, the keep-trace gradient descent is implemented by creating computation graphs for $g_t^{k(i)}$ during the automatic differentiation using Equation (7). We compare the proposed methods with the vanilla FedAvg [7] and the data sharing strategy (denoted as FedShare in the rest of this paper) proposed in [10].

The convergence curves of the proposed methods over FedShare and FedAvg in different FL settings are illustrated in Figure 1. As shown in the figure, both UGA and FedMeta outperform FedShare and

[1]FEMNIST and Shakespeare are from LEAF [2]

FedAvg in the convergence speed as well as the final rate of accuracy with a large margin. When combined together, FedMeta w/ UGA shows further improvements. More experimental results are available in the Appendix B.

## Acknowledgement

This work is done when the first author visits Youtu X-Lab, Tencent, as a research intern. This work is supported by the National Key R&D Program of China (2018YFB1003703), the National Natural Science Foundation of China (61936011), as well as the Beijing Key Lab of Networked Multimedia (Z161100005016051).

## References

[1] J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

[2] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[3] I. Chen, F. D. Johansson, and D. Sontag. Why is my classifier discriminatory? In *Advances in Neural Information Processing Systems*, pages 3539–3550, 2018.

[4] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[5] J. Konečnỳ, B. McMahan, and D. Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[6] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[8] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625, 2019.

[9] A. Olteanu, C. Castillo, F. Diaz, and E. Kiciman. Social data: Biases, methodological pitfalls, and ethical boundaries. *Frontiers in Big Data*, 2:13, 2019.

[10] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[11] J. Zou and L. Schiebinger. Ai can be sexist and racist-it's time to make it fair. *Nature*, 559(7714):324, 2018.

# Appendix

## A Pseudo-code

**Algorithm 1** Unbiased Gradient Aggregation

**Server Executes:**

Initialize $\omega_0$
**for** each round $t = 0, 1, ...$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of $m$ clients)
  **for** each client $k \in S_t$ **do in parallel**
    $g_t^k \leftarrow$ **ClientUpdate**$(k, \omega_t)$
  **end for**
  $\omega_{t+1} \leftarrow \omega_t - \eta_g \sum_{k \in S_t} \frac{n_k}{n_{S_t}} g_t^k$ ▷ *Equation (9)*
**end for**

**ClientUpdate**$(k, \omega_t)$**:** ▷ *Run on client* $k$

**for** $i$ in the total step of the first $E - 1$ epochs **do**
  $\omega_t^{k(i)} \leftarrow \omega_t^{k(i-1)} - \eta g_t^{k(i)}$ ▷ *Keep-trace gradient descent with Equation (7)*
**end for**
$g_t^k = \nabla_{\omega_t} \mathcal{L}(\omega_t^k; \mathcal{D}_k)$ ▷ *Equation (8)*
return $g_t^k$ to server

**Algorithm 2** Meta Updating

**Server Executes:**

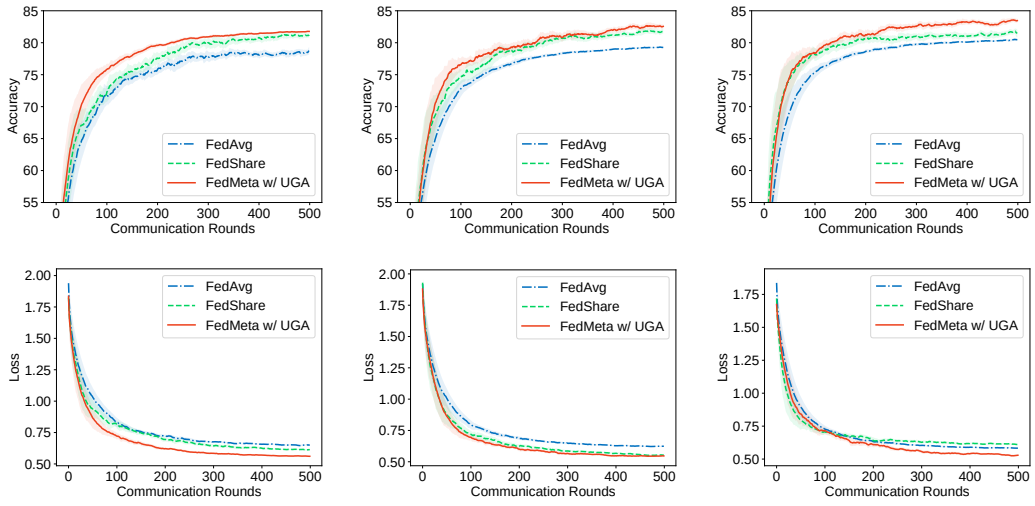
Initialize $\omega_0$
**for** each round $t = 0, 1, ...$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of $m$ clients)
  **for** each client $k \in S_t$ **do in parallel**
    $g_t^k \leftarrow$ **ClientUpdate**$(k, \omega_t)$ ▷ *Compatible with both FedAvg and Algorithm 1*
  **end for**
  $\omega_{t+1} \leftarrow \omega_t - \eta_g \sum_{k \in S_t} \frac{n_k}{n_{S_t}} g_t^k$ ▷ *Equation (9)*
  $\omega_{t+1}^{meta} = \omega_{t+1} - \eta_{meta} \nabla_{\omega_{t+1}} \mathcal{L}(\omega_{t+1}; \mathcal{D}_{meta})$ ▷ *Equation (10)*
**end for**

## B More Experimental Results

Table 1: Number of communication rounds to reach accuracy milestones & the convergence accuracy for all the methods on FEMNIST (non-IID) ($E = 5, B = 64$).

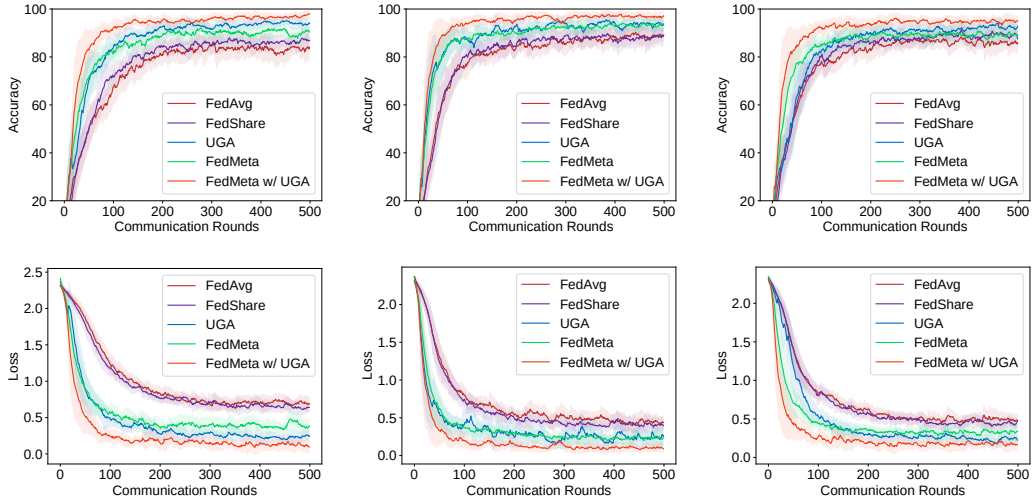
| Methods | Communication Rounds | | | Convergence Accuracy |
|---|---|---|---|---|
| | 70% | 80% | 90% | |
| FedAvg | 68 | 111 | 437 | 90.22 |
| FedShare | 65 | 93 | 385 | 90.74 |
| UGA | 27 | 48 | 137 | 95.87 |
| FedMeta | 30 | 49 | 155 | 94.98 |
| FedMeta w/ UGA | **21** | **31** | **59** | **98.18** |

* Bold fonts indicate better performances, i.e., fewer communication rounds and higher accuracy.




(a) $E = 2, B = 64$ (b) $E = 2, B = 128$ (c) $E = 5, B = 128$

Figure 2: Test accuracy (upper row) and loss (lower row) over communication rounds of different methods with IID data partitions on split CIFAR-10. (Better viewed in color)




(a) $E = 2, B = 64$ (b) $E = 5, B = 64$ (c) $E = 5, B = 128$

Figure 3: Test accuracy (upper row) and loss (lower row) over communication rounds of different methods with non-IID data partitions on FEMNIST. (Better viewed in color)

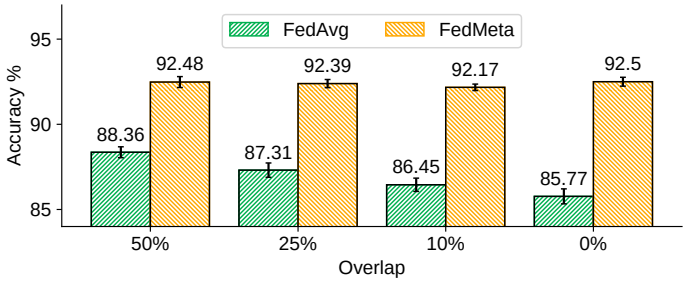



Figure 4: Test accuracy with different overlap rates of writers in FEMNIST between $\mathcal{D}_{meta}$ and $\mathcal{D}$. (Better viewed in color)