PERSONALIZED FEDERATED LEARNING WITH FIRST ORDER MODEL OPTIMIZATION.[FEDFOMO]

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) is an increasingly popular framework that traditionally aims for training of a single global model across decentralized local datasets. However, a single global model might not be ideal for all clients participating in the federation. In this work, we propose an alternative method, where each client federates with only the relevant other clients in order to obtain a stronger model on a client-specific objective. To achieve this personalization, rather than computing a single model average with constant weights for the entire federation as in traditional FL, we propose a new FL method that efficiently calculates an optimal model combination weight for each client. This weighting is calculated by interpreting other client models and computing how much a client can benefit from a list of available models. Furthermore, compared to previously proposed methods, we do not assume knowledge of any underlying data distributions or client similarities, and we allow each client to optimize for arbitrary target distributions of interest. This flexibility in arbitrary target optimization is a crucial step for personalization in a large federation. We evaluate our method on a variety of client configurations and degrees of local data heterogeneity with real world datasets. Our method outperforms existing relevant alternatives, while also enabling new features for personalized FL such as optimization outside of local data distributions.

1 INTRODUCTION

Federated learning (FL) has shown great promise in recent years for training a single global model over decentralized data. While the seminal motivation introduced FL for effective inference on a general test set similar in distribution to the clients' decentralized data in aggregate (McMahan et al., 2016; Bonawitz et al., 2019), here we focus on federated learning from a *client-centric* or *personalized* perspective. Our goal is then to enable stronger performance on personalized target distributions for each participating client. Such settings can be motivated by *cross-silo* FL, where the clients are autonomous data vendors (e.g. hospitals managing patient data, or corporations carrying customer information) that wish to collaborate without sharing private data (Kairouz et al., 2019). Instead of merely being a source of data and model training for the global server, clients may then take on a more active role, where their federated participation may be contingent on satisfying these client-specific target tasks and distributions. A strong FL framework in practice would then be able to flexibly accommodate these objectives, such that clients could optimize for arbitrary distributions simultaneously in a single federation.

In this setting, FL's realistic lack of an independent and identically distributed (IID) data assumption across clients may be both a burden and a blessing. As addressed in past work, learning a single global model across non-IID data batches can pose challenges such as non-guaranteed convergence and model parameter divergence (Hsieh et al., 2019; Zhao et al., 2018; Sahu et al., 2018). Furthermore, trying to fine-tune these global models may result in poor adaptation to local client test sets (Jiang et al., 2019). However, the non-IID nature of each client's local data can also provide useful signal for distinguishing their underlying local data distributions, without sharing any data.

We leverage this signal to propose a new framework for personalized federated learning. Our underlying method and its motivation are intuitively simplified as following: instead of giving all clients the same global model average weighted by a constant local training size as in prior work (McMa-

han et al., 2016), for each client we compute a weighted combination of the available models to best align with that client's interests, modeled by evaluation on a personalized target test distribution.

The key idea to enable this is that after each federating round, when clients send back their models we maintain these parameters individually, allowing clients in the next round to download a subset of them independently. Each federated update is then a two-step process, where clients (1) given a local objective clients their received models and (2) use these respective performances to weight each model's parameters in a personalized update. We show that this intuitive process can be thought of as a particularly coarse version of popular iterative optimization algorithms such as SGD, where instead of directly accessing other clients' data points and iteratively train our model parameters with the granularity of gradient decent, we limit ourselves to working with their uploaded models. We hence propose an efficient method to calculate these optimal combinations for each client. We call it `FedFomo`, as (1) each client's federated update is calculated with a simple first-order model optimization approximating a personalized gradient step, and (2) it draws inspiration from the "fear of missing out", every client no longer necessarily factors in contributions from all active clients during each federation round. In other words, curiosity *can* kill the cat. We therefore wish to save each model's personalized performance by restricting unhelpful models from each federated update.

We evaluate our proposed method with federated image classification and show that it outperforms other methods in a variety of non-IID scenarios. Furthermore, we demonstrate that because we compute federated updates directly with respect to client-specified local objectives, our framework can also optimize for *out-of-distribution* performance, where client's target distributions are different from their local training ones. In contrast, previous work that seeks to personalize based on similarity to a client's own model parameters (Mansour et al., 2020; Sattler et al., 2020) restricts this optimization to the local data distribution. We thus enable new features in personalized FL, and empirically demonstrate up to 70% improvement in some settings. Notably, we observe larger gains as the number of clients or level of non-IIDness increases.

Our contributions

1. We propose a flexible federated learning framework that allows clients to personalize to specific target data distributions irrespective of their available local training data.
2. Within this framework, we introduce a method to efficiently calculate the optimal weighted combination of uploaded models as a personalized federated update
3. Our method strongly outperforms other methods in non-IID federated learning settings.

2 Related Work

Federated Learning with Non-IID Data Federated learning traditionally trains a single global model across several decentralized data sources, aiming to perform well on a "global" data distribution similar in representation to an aggregation of each participating client's local data. In Federated Averaging (`FedAvg`) (McMahan et al., 2016), this is done through a simple dataset-weighted average of several local models, obtained by downloading a shared global model and training on locally available data for each active client. While fine-tuning this global model on a client's local data is a natural extension to achieve personalization (Mansour et al., 2020; Wang et al., 2019), prior work has shown that non-IID decentralized data can introduce challenges such as parameter divergence (Zhao et al., 2018), additional data distribution biases (Hsieh et al., 2019), and convergence is not always guaranteed Sahu et al. (2018).

Several more recent methods try to improve the robustness of global models under heavily non-IID datasets. `FedProx` (Sahu et al., 2018) adds a proximal term to the local model training objective such that the updated parameter values are kept close to the originally downloaded global model. This serves to reduce the potential weight divergence defined in Zhao et al. (2018), who instead tackle the problem by allowing each client to share a small subset of its own data with every other client. This effectively makes each client's local training set closer in distribution to the global test set. More recently, Hsu et al. (2019) propose to add momentum to the global model update in `FedAvgM` to reduce the possibly harmful oscillations associated with averaging local models after several rounds of stochastic gradient descent for non-identically distributed data.

While these advances may make training a global model more robust across non-IID local data sources, they do not directly address performance on local-level data distributions relevant to individual clients. As argued by Jiang et al. (2019), this latter task may be more naturally desired in any FL scenario with non-IID data, as differences in local training data may suggest that only a subset of all potential features or outcomes are relevant to each client. Such target distributions may be fairly different from the global aggregate, especially in highly personalized scenarios, and the resulting dataset shift may be difficult to handle with a single model. We avoid this issue by concurrently training models that optimize directly for these target distributions.

Personalized Federated Learning Given the challenges above with learning a single global model, other proposed approaches aim to train multiple models or personalizing components as a means to deal with multiple target distributions. Smith et al. (2017) propose a multi-task learning solution to FL with `MOCHA`, a distributed MTL framework that approaches each client as a task and learns a single model for each client. Mixture methods (Deng et al., 2020; Hanzely & Richtárik, 2020; Mansour et al., 2020) aim to compute a personalized combination of model parameters between a client's local model and the current global model. Meanwhile, Liang et al. (2020) instead apply this mixing across network layers, where the lower layers act as local-specific encoders that map a client's observed data to input for a globally shared classifier. Instead of only mixing with a shared global model, our work allows for greater control and distinct mixing parameters with multiple local models. Finally, work in *clustered federated learning* (Sattler et al., 2020; Ghosh et al., 2020; Briggs et al., 2020; Mansour et al., 2020) assume that there exist inherent partitions or data distributions that govern clients' local data, and personalize by clustering clients based on the distances between their local model parameters as a similarity metrics. In doing so, they aim to discover these partitions, and subsequently learn federated models within each cluster. One key difference is that we do not restrict which models are computed together in a federation, allowing any client to download a suitable model. We also adjust the actual weights in each model average, achieving a greater degree of personalization. Unlike any approach before, we are also not limited to optimizing for a test distribution that is the same as client's local training data. Accordingly, we are the first to our knowledge to enable transfer to new specific data domains of interest through personalized FL.

3 Federated First Order Model Optimization

We now present `FedFomo`, a personalized federated learning framework that aims to efficiently compute client-optimizing federated updates. Here we adopt the general structure of most FL methods, where an iterative training procedure cycles between downloading model parameters from a server to each active client, training the models locally on each client's data, and sending back the updated models to the server for future rounds. However, as we do not compute a single global model, each federated download introduces two new steps: (1) figuring out which models to send to which clients, and (2) computing their personalized weighted combinations for each client. We define our problem and notation describe how we accomplish (1) and (2) in the following sections.

Problem Definition and Notation Our work most naturally applies to heterogeneous federated settings where we critically do not restrict the participating clients to a single distribution local training or target test distribution. We then wish to satisfy all participating clients simultaneously without any additional prior knowledge.

Let $\mathbb{C}$ be a population with $|\mathbb{C}| = K$ total clients, where each client $c_i \in \mathbb{C}$ carries local data $\boldsymbol{D}_i$ sampled from some distribution $\mathcal{D}_j$ and local model parameters $\theta_i^{\ell(t)}$ during any round t. Each c_i also maintains some personalized objective or task $\mathcal{T}_i$ motivating their participation in the federation. Here we focus on supervised classification as a universal task setting. Each client and task are then associated with a test dataset $\boldsymbol{D}_i^{\text{test}} \sim \mathcal{D}_j^*$. We define each $\mathcal{T}_i := \min \mathcal{L}(\theta_i^{\ell(t)}; \boldsymbol{D}_i^{\text{test}})$, where $\mathcal{L}(\theta; \boldsymbol{D}) : \Theta \mapsto \mathbb{R}$ is the loss function associated with dataset $\boldsymbol{D}$, and Θ denotes the space of models possible with our presumed network architecture. We assume no knowledge regarding clients and their data distributions, nor that test and local data belong to the same distribution. We aim to obtain the optimal set of model parameters $\{\theta_1^*, \dots, \theta_K^*\} = \arg\min \sum_{i \in [K]} \mathcal{L}_{\mathcal{T}_i}(\theta_i)$.

3.1 COMPUTING FEDERATED UPDATES WITH FOMO

Unlike previous work in federated learning, `FedFomo` learns to simultaneously compute optimal weighted combinations of the available server models for each participating client. To do so, we leverage information from clients in two different ways. First, we aim to directly optimize for each client's target objective. To do so, we assume that clients can distinguish between good and bad models on their target tasks, through the use of a labeled validation data split $\boldsymbol{D}_i^{\text{val}} \subset \boldsymbol{D}_i$ in the client's local data. $\boldsymbol{D}_i^{\text{val}}$ should be similar in distribution to the target test dataset $\boldsymbol{D}_i^{\text{test}}$. The client can then evaluate any arbitrary model θ_j on this validation set, and quantify the performance through the computed loss which we denote by $\mathcal{L}_i(\theta_j)$.

Second, we directly leverage the potential heterogeneity among client models. Zhao et al. (2018) explore this phenomenon as a failure mode for traditional single model FL, where they show that diverging model weights come directly from local data non-IIDness. However, instead of combining these parameters into a single global model, we maintain the uploaded client models individually. Federated updates for each client then involve an initial model download followed by evaluation on their target distributions through their aforementioned validation splits.

Critically, these two ideas together not only allow us to compute more personal model updates within non-IID local data distributions, but also enable clients to optimize for data distributions different from their own local data's. We now present our personalized federated update method.

Federated learning as an iterative local model update The central premise of our work stems from viewing each federated model download−and subsequent changing of local model parameters−as an optimization step towards some objective. In traditional FL, this objective involves performing well on the global population distribution, similar in representation to the union of all local datasets. Assuming N federating clients, we compute each global model θ^G at time t as:

$$\theta^{G(t)} = \sum_{n=1}^{N} w_n \cdot \theta_n^{\ell(t)}, \text{ where } w_n = \frac{|\boldsymbol{D}_n^{\text{train}}|}{\sum_{j=1}^{N} |\boldsymbol{D}_j^{\text{train}}|} \tag{1}$$

If client c_i downloads this model, we can view this change to their local model as an update:

$$\theta_i^{\ell(t+1)} \leftarrow \theta_i^{\ell(t)} + \sum_{n=1}^{N} w_n \cdot \left(\theta_n^{\ell(t)} - \theta_i^{\ell(t)}\right) \tag{2}$$

since as defined in (1), $\sum_n w_n = 1$. We see that downloading the global model subsequently updates a client's current local model parameters in directions specified by the weights $\boldsymbol{w}$ and the other models $\{\theta_n\}$ in the federation. A natural choice to optimize for the global target distribution is then to set w_n as in Eq 1 and McMahan et al. (2017), e.g. as an unbiased estimate of global model parameters. However, in our personalized scenario, we are more interested in computing the update uniquely with respect to each client's target task. We then wish to find the optimal weights $\boldsymbol{w} = \langle w_1, \ldots, w_N \rangle$ that optimize for the client's objective, minimizing $\mathcal{L}_{V_i}(\theta_i^{\ell})$.

Efficient personalized FL with FOMO Intuitively speaking, we want to identify the other models $\{\theta_m^{\ell(t)} : m \in [N] \setminus i\}$ where adopting or moving towards their parameter values leads to better performance of our model towards our target distribution, and accordingly weight them higher in a model average. Recall that if a client carries a satisfying number of local data points associated with their target objective $\mathcal{L}_i$, then they could obtain a reasonable model through local training alone, e.g. directly updating their model parameters through several rounds of SGD:

$$\theta_i^{\ell(t+1)} \leftarrow \theta_i^{\ell(t)} - \alpha \nabla_\theta \mathcal{L}_i(\theta_i^{\ell(t)}) \tag{3}$$

However, without this data, clients are more motivated to federate. In doing so they obtain useful updates, albeit in the more restricted form of fixed model parameters $\{\theta_n : n \in N\}$ (Eq. 7). Then for personalized or non-IID target distribution settings, we can iteratively solve for the optimal combination of client models $\boldsymbol{w}^* = \arg\min \mathcal{L}_{V_i}(\theta)$ by computing:

$$\theta_i^{\ell(t+1)} \leftarrow \theta_i^{\ell(t)} - \alpha \nabla_{\boldsymbol{w}} \mathcal{L}_i(\theta_i^{\ell(t)}) \tag{4}$$

Unfortunately, as the larger federated learning algorithm is already an iterative process with many rounds of communication, computing $\boldsymbol{w}^*$ through Eq. 4 may be a cumbersome procedure. Worse, if the model averages are only computed server-side as in traditional FL, then Eq. 4 becomes prohibitively expensive in communication rounds (McMahan et al., 2017).

Following this line of reasoning however, we thus derive an approximation of $\boldsymbol{w}^*$ for any client: Given previous local model parameters $\theta_i^{\ell(t-1)}$, set of fellow federating models available to download $\{\theta_n^{\ell(t)}\}$, local training learning rate α, and local client objective captured by $\mathcal{L}_i$, we propose weights of the form:

$$w_n = \frac{\mathcal{L}_i(\theta_i^{\ell(t-1)}) - \mathcal{L}_i(\theta_n^{\ell(t)})}{\theta_n^{\ell(t)} - \theta_i^{\ell(t-1)}} \tag{5}$$

such that the resulting federated update

$$\theta_i^{\ell(t+1)} \leftarrow \theta_i^{\ell(t+1)} + \sum_{n \in [N]} w_n(\theta_n^{\ell(t)} - \theta_i^{\ell(t-1)}) \tag{6}$$

directly optimizes for client c_i's objective up to a first-order approximation.

In practice, we normalize with $w_n = \dfrac{\max(w_n, 0)}{\sum_{n=1}^{N} \max(w_n, 0)}$ to maintain $w \in [0, 1]$ and $\sum_{n=1} w_n = 1$.

We derive Eq. 5 in Appendix A.1, but note here that our formulation captures the intuition that we should federate with client models that perform better than our own model, e.g. have a smaller loss on $\mathcal{L}_i$. Moreso, we weigh models more heavily as this positive loss delta increases, or the distance between our current parameters and theirs decreases, in essence most heavily weighing the models that most efficiently improve our performance. We use parameters at $t-1$ so we can directly compute how much we should weigh our current parameters $\theta_i^{\ell(t)}$ in the federated update.

Learning with limited model download bandwidth Realistically, `FedFomo` needs to consider the additional bandwidth from downloading multiple models. While we can alleviate this with techniques in network quantization and distillation (Chen et al., 2017; Hinton et al., 2015; Xu et al., 2018), in practice `FedFomo` also figures out which models benefit which clients over time to only subsequently send those models to their respective clients in future federating rounds.

To do so, we model these client-to-client "preferences" as an affinity matrix $\boldsymbol{P}$ composed of vectors $\boldsymbol{p}_i = \langle p_{i,1}, \ldots, p_{i,K} \rangle$ expressing each client c_i's likelihood with federating with any other client. We do not strictly maintain these values as probabilities, but only consider the relative values as an ordered ranking. For each active client we query its associated preference vector for each of N uploaded models from the previous round, and use an ε-greedy sampling strategy to obtain the models for each client to download. We update $\boldsymbol{p}$ using the unclipped and un-normalized model weights computed in Eq. 5, which requires clients to report this scalar back to the server. We initialize $\boldsymbol{P} = \text{diag}(1, \ldots, 1)$, such that over time we discover better performing models for each client and encourage subsequent downloads.

4 EXPERIMENTS

Experimental Setup We consider two different scenarios for simulating non-identical data distributions across federating clients. As a connection to prior work, we evaluate with the *pathological* non-IID data setup presented in McMahan et al. (2016), where each client is randomly assigned two classes among N total classes. We also utilize a *latent distribution* non-IID setup, where we first partition our datasets based on feature and semantic similarity, and then sample from them to setup different local client data distributions. We describe further details of our setup in Appendix A.2. We evaluate under both setups with two FL scenarios: 15 and 100 clients with 100% and 10% participation respectively. Finally we report final accuracy after training with $E = 5$ local epochs per round for 20 communication rounds in the former and 100 rounds in the latter.

Dataset and Model Implementation Details We evaluate on four image classification datasets. Based on prior work (McMahan et al., 2016; Liang et al., 2020), we use the MNIST (LeCun et al.,

	MNIST		CIFAR-10	
	15 clients	100 clients	15 clients	100 clients
Local Training	99.62 ± 0.21	94.25 ± 4.28	92.73 ± 0.87	85.42 ± 4.06
`FedAvg` (McMahan et al., 2016)	91.97 ± 2.17	78.83 ± 9.68	57.12 ± 1.14	53.08 ± 7.40
`FedAvg` + Data (Zhao et al., 2018)	91.99 ± 2.14	78.85 ± 9.66	58.5 ± 1.67	56.62 ± 8.92
`FedProx` (Sahu et al., 2018)	90.94 ± 1.85	79.06 ± 10.88	54.60 ± 3.26	52.92 ± 5.56
`LG-FedAvg` (Liang et al., 2020)	$\mathbf{99.79 \pm 0.07}$	84.17 ± 1.92	92.36 ± 1.00	84.17 ± 4.45
`MOCHA` (Smith et al., 2017)	94.74 ± 2.27	84.58 ± 5.80	$\mathbf{93.85 \pm 2.04}$	76.09 ± 8.49
Ours n=5	99.62 ± 2.91	$\mathbf{98.81 \pm 1.26}$	93.01 ± 0.96	92.10 ± 5.20
Ours n=10	99.63 ± 0.07	98.71 ± 2.86	92.73 ± 0.96	$\mathbf{92.67 \pm 4.21}$

Table 1: Personalized federated learning accuracy with pathological non-IID splits. Best results in bold. Across pathological non-IID settings, our method outperforms or achieves competitive personalization accuracy, especially with larger federating populations.

1998), CIFAR-10 (Krizhevsky et al., 2009) datasets, and for more challenging versions also consider the EMNIST (Cohen et al., 2017) and CIFAR-100 datasets. For all experiments, we use the same CNN model architecture in McMahan et al. (2016). We train with SGD, lr$= 0.1$, 0 momentum, 1e-4 weight decay, and 0.99 learning rate decay for CIFAR-10/100, and change learning rate to 0.01 for MNIST and EMNIST. For `FedFomo` we report results with $n = 5$ and $n = 10$ model downloads each round, and separate $\boldsymbol{D}^{\text{train}}$ and $\boldsymbol{D}^{\text{val}}$ with an 80-20 split for initially allocated local train data.

Latent Distribution Non-IID setup We organize our data into latent distributions with total number $\in \{1, 2, 3, 4, 5, 10\}$ (the foremost being equivalent to the IID setting), and report the average Earth Mover's Distance (EMD) between local client data and the total dataset across all clients to quantify non-IIDness. We evenly allocate clients among distributions

Federated Learning Baselines We compare `FedFomo` against a variety of existing federated learning frameworks.We also compare with related work that deal with highly heterogeneous datasets and personalized evaluation. These methods broadly fall under two categories: (1) they propose modifications to training a single global model that make the global model more robust to non-IID local datasets, or (2) aim to train more than one model or model component to personalize performance directly to client test sets. For the former group, we consider `FedAvg`, `FedProx`, and the 5% data-sharing strategy with `FedAvg`, while in the latter we compare our method to `LG-FedAvg`, `MOCHA`, and a local training baseline. All accuracy are reported with mean and standard deviation over three runs, with local training epochs $E = 5$, the same number of communication rounds (20 for 15 clients, 100% participation; 100 for 100 clients, 10% participation) and learning rate 0.01 for MNIST, 0.1 for CIFAR-10). We implemented all results, using code from https://github.com/pliang279/LG-FedAvg for `LG-FedAvg` and `MOCHA`.

4.1 In-distribution Personalized Evaluation in non-IID Federated Learning

We first evaluate the in-distribution personalization performance in both *pathological* and *latent distribution* settings, where each client optimizes performance on target data distribution that is similar in distribution to its local training set. Such scenarios are common in settings where no client can directly access enough data to train an adequately performing model in the first place.

Pathological Non-IID We follow precedent and report accuracy after assigning two classes out of the ten to each client for the pathological setting in Table 1. Across datasets and client setups, our proposed `FedFomo` strongly outperforms alternative methods in settings with larger number clients, and achieves competitive accuracy in the 15 client scenario. In the larger 100 client scenario, each individual client participates less frequently but also carries less local training data. Such settings motivate a higher demand for efficient federated

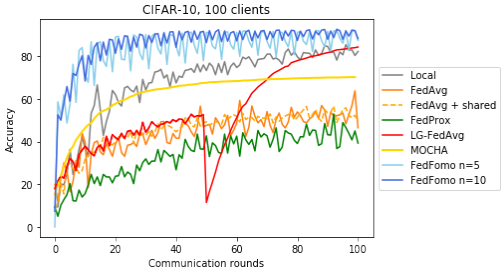


Figure 1: non-IID client convergence.

updates, as there are less training rounds for each client overall. Meanwhile, methods that try to train a single robust model perform with mixed success over the `FedAvg` baseline, and notably do not perform better than local training alone. Despite the competitive performance, we note that this pathological setting is not the most natural scenario to apply `FedFomo`. In particular when there are less clients, each client's target distribution carries only 2 random classes, there is no guarantee that any two clients share the same objective such that they can clearly benefit each other. With more clients however, we can also expect higher frequencies of target distribution overlap, and accordingly find that we outperform all other methods. In Fig. 1, we further support this as our method is able to more quickly converge to higher accuracies in the 100-client scenario.

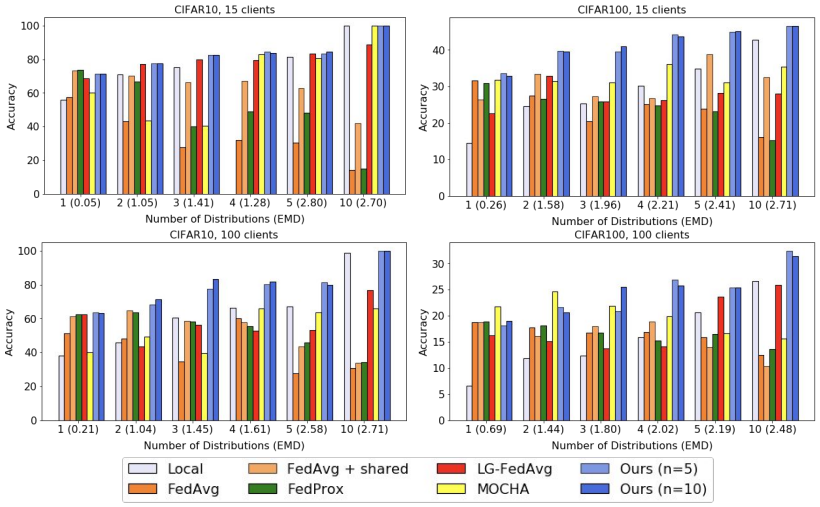


Figure 2: Classification accuracy of FL frameworks over datasets and latent distributions.

Latent Distribution Non-IID We next report how each FL method performs in the latent distribution setting in Fig. 2. Here we study the relative performance of `FedFomo` across various levels of statistical heterogeneity, and again show that our method is able to strongly outperform others in highly non-IID settings. We observe this performance gap widen as local datasets become more non-IID, where global FL methods may suffer more from combining increasingly divergent weights while also experiencing high target data distribution shift (quantified with higher EMD) due to local data heterogeneity. Sharing a small amount of data among clients uniformly helps, as does actively trying to reduce this divergence through `FedProx`, but higher performance most convincingly come from methods that do not rely on a single model. The opposite trend occurs with local training, as more distributions using the same 10 or 100 classes leads to smaller within-distribution variance. Critically, `FedFomo` is competitive with local training in the most extreme non-IID case while strongly outperforming `FedAvg`, and outperforms both in moderately non-IID settings (EMD $\in [1, 2]$), suggesting that we can selectively leverage model updates that best fit client objectives to justify federating. When data is more IID (EMD $\in (0, 1]$), it is more likely that any individual client models can benefit each other, and it becomes harder for a selective update to beat a general model average. `FedFomo` also outperforms personalizing-component and multi-model approaches (`MOCHA` and `LG-FedAvg`), where with respect to data heterogeneity we see similar albeit weaker and more stochastic trends in performance.

4.2 Out-of-Distribution Target Test Evaluation

To further evaluate the flexibility of our method, we further consider the non-IID federated setting where each client optimizes for a target distribution which is not the same as their local training data distribution. In this scenario, although a client may carry enough local data to train an adequate model for one domain, it may have another target data distribution of interest where the relevant data points are hard to access. For example, in a self-driving scenario, an individual client might not

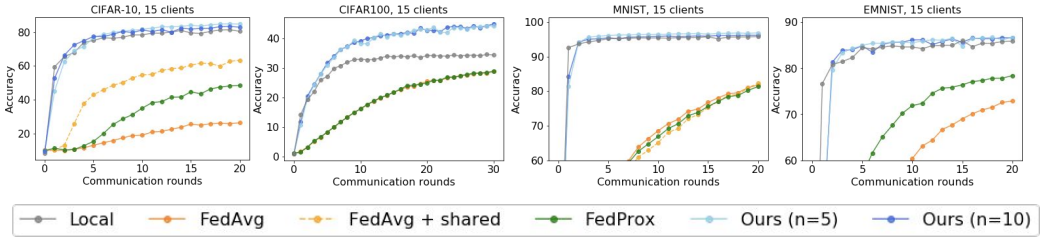


Figure 3: Accuracy on image classification datasets with 15 clients and 100% participation.

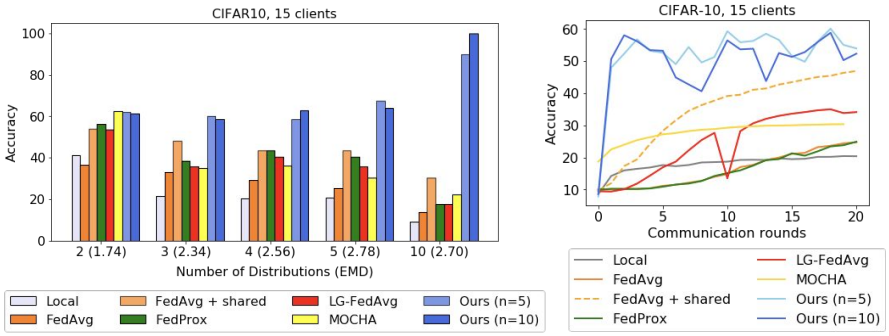


Figure 4: Federated evaluation on out-of-local-distribution data. **Left**: Highest accuracy across increasingly non-IID settings. **Right**: 20 communication round accuracy with 5 latent distributions.

have access to enough data for certain classrd due to geographical constraints. However, for safety during deployment, it needs to leverage this information from others. To simulate this scenario, after organizing data into latent distributions we allow clients to optimize for another single distribution by randomly shuffling $(\boldsymbol{D}^{\text{val}}, \boldsymbol{D}^{\text{test}})$ as a pair among clients. We test on the CIFAR-10 dataset with 15 clients and full participation per round. We repeat this shuffling procedure five times, and report the mean accuracy over all runs and clients.

Local out-of-distribution As shown in Fig. 4.2, our method is able to consistently strongly outperform alternatives with up to a 70% improvement in accuracy in non-IID settings. In all settings we compare methods using the same train and test splits randomly shuffled between clients, such that through shuffling we encounter potentially large amounts of data variation between a client's training data and its test set. This then supports the validity of the validation split and downloaded model evaluation components in our method to uniquely optimize for arbitrary data distributions different from a client's local training data. On the other hand all methods other than ours are unable to convincingly handle optimizing for a target distribution that is different from the client's initially assigned local training data. Sharing data expectedly stands out among other methods that do not directly optimize for a client's objective, as each client then increases the label representation overlap between its train and test sets. We note that in the 2-distribution setting, where each client's training data consists of 5 classes on average, the higher performance of other methods is most likely an result of our simulation, where if we only have two distributions to shuffle between there is a greater chance that more clients end up with the same test distribution. Our method consistently achieves significant improvements over others in this local out-of-distribution evaluation setting.

5 Conclusion

We present `FedFomo`, a flexible federated learning framework that achieves strong performance in personalized FL across various non-IID settings, and uniquely enables clients to also optimize for target distributions distinct from their local training data.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. *arXiv preprint arXiv:2004.11791*, 2020.

Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, pp. 742–751, 2017.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017. URL `http://arxiv.org/abs/1702.05373`.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*, 2020.

Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B Gibbons. The non-iid data quagmire of decentralized machine learning. *arXiv preprint arXiv:1910.00189*, 2019.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Paul Pu Liang, Terrance Liu, Liu Ziyin, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016. URL http://arxiv.org/abs/1602.05629.

Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018. URL http://arxiv.org/abs/1812.06127.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *CoRR*, abs/1705.10467, 2017. URL http://arxiv.org/abs/1705.10467.

Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. Deep neural network compression with single and multiple level quantization. *arXiv preprint arXiv:1803.03289*, 2018.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018. URL http://arxiv.org/abs/1806.00582.

A Appendix

A.1 Deriving the FOMO Update

Recall that we can view each federated model download can be viewed as an iterative update,

$$\theta_i^{\ell(t+1)} = \theta_i^{\ell(t)} + \sum_{n=1}^{N} w_n \cdot \left(\theta_n^{\ell(t)} - \theta_i^{\ell(t)}\right) \tag{7}$$

where given a client's current parameters $\theta_i^{\ell(t)}$, the weights $\boldsymbol{w} = \langle w_1, \ldots, w_N \rangle$ in conjunction with the model deltas $\theta_n^{\ell(t)} - \theta_i^{\ell(t)}$ determine how much each client should move its local model parameters to optimize for some objective. Unlike more common methods in machine learning such as gradient descent, the paths we can take to get to this objective are restricted by the fixed model parameters $\{\theta_n^{\ell(t)}\}$ available to us at time t. While traditional FL methods presume this objective to be global test set performance, from a client-centric perspective we should be able to set this objective with respect to any dataset or target distribution of interest.

We then view this problem as a constrained optimization problem where $\sum_{n \in [N]} = 1$. As a small discrepancy, if $i \in [N]$, then to also calculate w_i or how much client c_i should weigh its own model in the federated update directly, we reparameterize Eq. 7 as an update from a version of the local model *prior* to its current state, e.g.

$$\theta_i^{\ell(t+1)} = \theta_i^{\ell(t-1)} + \sum_{n=1}^{N} w_n \cdot \left(\theta_n^{\ell(t)} - \theta_i^{\ell(t-1)}\right) \tag{8}$$

and again we have current budget of 1 to allocate to all weights $\boldsymbol{w}$. Additionally, to go along with Eq. 8, we deviate a bit from the optimal $t+1$ term in Eq. 4 and set

$$\theta_i^{\ell(t+1)} = \theta_i^{\ell(t)} \leftarrow \theta_i^{\ell(t-1)} - \alpha \nabla_{\boldsymbol{w}} \mathcal{L}_i(\theta_i^{\ell(t-1)}) \tag{9}$$

There is then a parallel structure between Eq. 8 and Eq. 9, and we proceed by trying to find optimal $\boldsymbol{w}$ that closely approximate the gradient $\nabla_{\boldsymbol{w}}$.

For client c_i, we can use a Taylor series expansion to approximate the update $\nabla_{\boldsymbol{w}} \mathcal{L}_i(\theta_i^{\ell(t-1)})$:

$$\mathcal{L}_i(\theta_i^{\ell(t)}) = \mathcal{L}_i(\theta_i^{\ell(t-1)}) + \nabla_{\boldsymbol{w}} \mathcal{L}_i(\theta_i^{\ell(t-1)})^\top (\theta_i^{\ell(t)} - \theta_i^{\ell(t-1)}) + \dots \tag{10}$$

As an approximation, we consider up to the first order term and drop the rest. As $\theta_i^{\ell(t)} = \boldsymbol{w}^\top \theta_i^{\ell(t-1)}$, for each w_n we define a function:

$$\varphi(w_n) := w_n \cdot \theta_n^{\ell(t)} + (1 - w_n) \cdot \theta_i^{\ell(t-1)} \tag{11}$$

to parameterize the θ's as functions of our weights. For each partial in Eq. 10, we can re-express θ's as $\varphi(w_n)$'s, and get

$$\frac{\partial}{\partial w_n} \mathcal{L}_i(\theta_i^{\ell(t-1)}) = \mathcal{L}_i(\theta_n^{\ell(t)}) - \mathcal{L}_i(\theta_i^{\ell(t-1)}) \tag{12}$$

for any given w_n. Finally, to connect this to our estimated $\hat{\boldsymbol{w}}^*$, we note the equivalency between Eq. 8 and Eq. 9, where

$$\hat{\boldsymbol{w}}^\top \big(\theta_n^{\ell(t)} - \theta_i^{\ell(t-1)}\big) = -\alpha \Big\langle \mathcal{L}_i(\theta_1^{\ell(t)}) - \mathcal{L}_i(\theta_i^{\ell(t-1)}), \dots, \mathcal{L}_i(\theta_n^{\ell(t)}) - \mathcal{L}_i(\theta_i^{\ell(t-1)}) \Big\rangle \tag{13}$$

And so each individual weight for client c_i considering model θ_n can be computed as

$$w_n = -\alpha \cdot \frac{\mathcal{L}_i(\theta_n^{\ell(t)}) - \mathcal{L}_i(\theta_i^{\ell(t-1)})}{\theta_n^{\ell(t)} - \theta_i^{\ell(t-1)}} \tag{14}$$

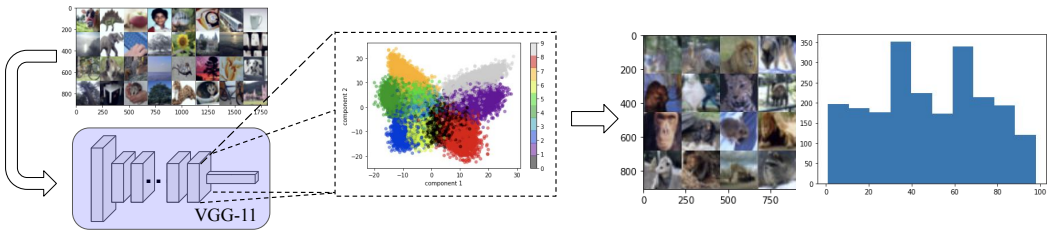
We arrive at Eq. 5 by distributing the negative α to capture the right direction in each update, and then dropping the term as a constant because in practice we normalize with

$$w_n = \frac{\max(w_n, 0)}{\sum_{n=1}^{N} \max(w_n, 0)} \tag{15}$$

A.2 Latent Distribution Non-IID Motivation and Setup

In this subsection, we discuss our latent distribution non-IID setting in more detail. We believe the pathological setup though useful might not represent more realistic or frequent occurring setups. As an example, a world-wide dataset of road landscapes may vary greatly across different data points, but variance in their feature representations can commonly be explained by their location. In another scenario, we can imagine that certain combinations of songs, or genres of music altogether are more likely to be liked by the same person than others. In fact, the very basis and success of popular recommender system algorithms such as collaborative filtering and latent factor models rely on this scenario (Hofmann, 2004). Accordingly, in this sense statistical heterogeneity and client local data non-IIDnes is more likely to happen in groups.

We thus propose and utilize a latent distribution method to evaluate `FedFomo` against other more recent proposed FL work. To use this setting, we first compute image representations by training a VGG-11 convolutional neural network to at least 85% classification accuracy on a corresponding dataset. We then run inference on every data point, and treat the 4096-dimensional vector produced in the second fully-connected layer as a semantic embedding for each individual image. After further reduction to 256 dimensions through PCA, we use K-Means clustering to partition our dataset into D disjoint distributions. Given K total clients, we then evenly assign each client to a distribution $\mathcal{D}$. For each client we finally obtain its local data by sampling randomly from $\mathcal{D}$ without replacement. For datasets with pre-defined train and test splits, we cluster embeddings from both at the same time such that similar images across splits are assigned the same K-means cluster, and respect these original splits such that all $\boldsymbol{D}^{\text{test}}$ images come from the original test split. (Fig. 5)



(a) Train image classification model to learn expressive representations

(b) Extract representative feature vectors and further reduce dimensionality with PCA

(b) Use K-Means clustering to partition into D natural groups with unique class distributions

Figure 5: Visual overview for generating latent distributions using image classification datasets.

A.3 Practical Considerations and Aspects of FedFomo

As previously stated in our proposed method FedFomo each federated update is a two-part process where we first send individual models to the client. Clients then follow our protocol in Section 3.1. In practice, we implement these interactions such that downloaded models are treated like black boxes, e.g. the actual weight values are not accessible to the client, and the client can only query the model running inference on its validation set. During each federated update, the clients also do not know whose model they download, and we can restrict any identifying connections between client and uploaded model to maintain reasonable levels of anonymity. If further privacy is desired, because each model query can be considered a post-processing step in data analysis (Dwork et al., 2014), we can obtain additional privacy guarantees through locally training with differential privacy as in Abadi et al. (2016).

A.4 Discussion and additional ablations

In this section we present additional work on key hyperparameters or aspects of FedFomo to give further insight into our method's functionality and robustness to parameters. We first consider key design choices related to the size of each client's validation split, and the number of models to download during each federated update.

Size of the validation split To better organize federated uploaded models into personalized federated updates, our method requires a local validation split $\boldsymbol{D}^{\text{val}}$ that reflects the client's objective or target test distribution. Here, given a pre-defined amount of locally available data, we ask the natural question of how a client should best go about dividing its data points between those to train its own local model and those to evaluate others with respect to computing a more informed personalized update through FedFomo. We use the 15 client 100% participation setup with 5 latent distributions organized over the CIFAR-10 dataset, and consider both the evaluation curve and final test accuracy over allocating a fraction $\in \{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 0.9\}$ of all clients' local data to $\boldsymbol{D}^{\text{val}}$, and track evaluation over 20 communication rounds with 5 epochs of local training per round. On average, each client has 3333 local data points. We denote final accuracy and standard deviation over five runs in Fig 6.

As reported in Fig. 6, we observe faster convergence to a higher accuracy when allocating under half of all local data points to the validation split, with a notable drop-off using more data points. This is most likely a result of reducing the amount of data available for each client to train their model locally. Eventually this stagnates, and observe a slight decrease in performance between validation split fraction 0.05 and 0.1.

Number of models to download As addressed in Section A.3, an open hyperparameter in FedFomo is the number of models we allow each client to download and subsequently evaluate when computing a personalized update. In previous experiments, we reported accuracies when downloading 5 or 10 models. While naturally being able to evaluate more models may presume a more optimal update approximation, including more models in each federated download may be costly from a communication perspective, and there may be diminishing returns to trying to factor in more models. There is also greater inference time required to evaluate more models. While a full

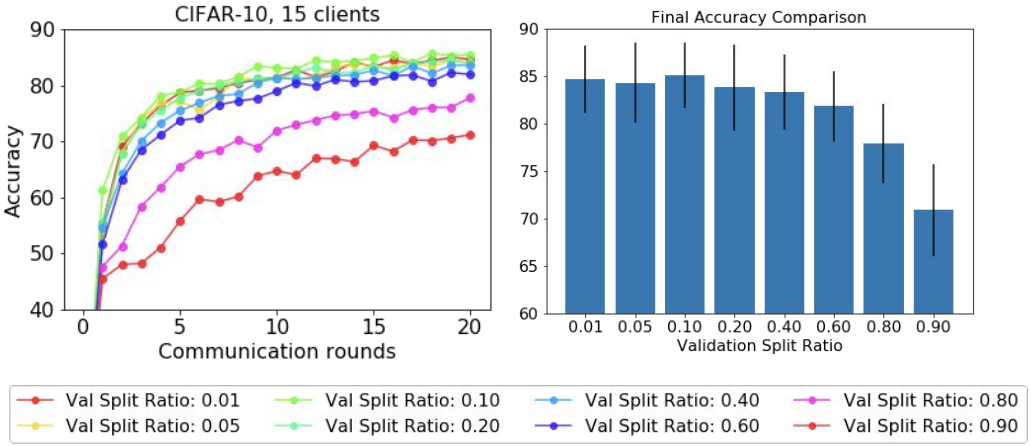


Figure 6: In-distribution accuracy over validation split ratio.

analysis on the communication costs running `FedFomo` is out of scope of this work, here we shed light on the optimal number of models to download. Again we consider the 15 client 5 latent distribution setup, and report evaluation over 20 communication rounds and $n \in \{1, 3, 5, 7, 9, 10, 14\}$ models to download on the CIFAR-10 and CIFAR-100 datasets in Fig. 7 across 5 seeds. We note that empirically any performance benefits from downloading more models appear to drop off after a certain point, with highest performance in between downloading 3 and 7 models.

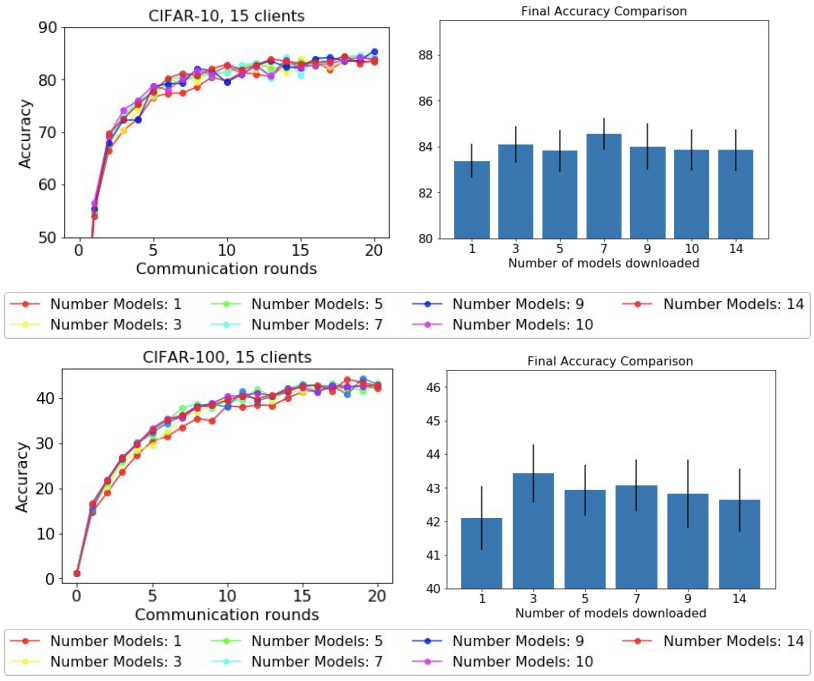


Figure 7: `FedFomo` number of clients per download ablation