Multi-Model Federated Learning with Provable Guarantees

Neelkamal Bhuyan[1(✉)], Sharayu Moharir[1], and Gauri Joshi[2]

[1] Indian Institute of Technology Bombay, Mumbai, India
neelkamalbhuyan@gmail.com
[2] Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. Federated Learning (FL) is a variant of distributed learning where edge devices collaborate to learn a model without sharing their data with the central server or each other. We refer to the process of training multiple independent models simultaneously in a federated setting using a common pool of clients as multi-model FL. In this work, we propose two variants of the popular FedAvg algorithm for multi-model FL, with provable convergence guarantees. We further show that for the same amount of computation, multi-model FL can have better performance than training each model separately. We supplement our theoretical results with experiments in strongly convex, convex, and non-convex settings.

Keywords: Federated Learning · Distributed Learning · Optimization

1 Introduction

Federated Learning (FL) is a form of distributed learning where training is done by edge devices (also called clients) using their local data without sharing their data with the central coordinator (also known as server) or other devices. The only information shared between the clients and the server is the update to the global model after local client training. This helps in preserving the privacy of the local client dataset. Additionally, training on the edge devices does not require datasets to be communicated from the clients to the server, lowering communication costs. The single model federated learning problem has been widely studied from various perspectives including optimizing the frequency of communication between the clients and the server [17], designing client selection algorithms in the setting where only a subset of devices train the model in each iteration [9], and measuring the effect of partial device participation on the convergence rate [12].

Closest to this work, in [2], the possibility of using federated learning to train multiple models simultaneously using the same set of edge devices has been explored. We refer to this as multi-model FL. In the setting considered in

This work is generously supported by NSF CNS #2112471.

E. Hyytiä and V. Kavitha (Eds.): VALUETOOLS 2022, LNICST 482, pp. 207–222, 2023.
https://doi.org/10.1007/978-3-031-31234-2_13

[2], each device can only train one model at a time. The algorithmic challenge is to determine which model each client will train in any given round. Simulations illustrate that multiple models can indeed be trained simultaneously without a sharp drop in accuracy by splitting the clients into subsets in each round and use one subset to train each of the models. One key limitation of [2] is the lack of analytical performance guarantees for the proposed algorithms. In this work, we address this limitation of [2] by proposing algorithms for multi-model FL with provable performance guarantees.

1.1 Our Contributions

In this work, we focus on the task of training M models simultaneously using a shared set of clients. We propose two variants of the Fed-Avg algorithm [15] for the multi-model setting, with provable convergence guarantees. The first variant called Multi-FedAvg-Random (MFA-Rand) partitions clients into M groups in each round and matches the M groups to the M models in a uniformly randomized fashion. Under the second variant called Multi-FedAvg-Round Robin (MFA-RR), time is divided into frames such that each frame consists of M rounds. At the beginning of each frame, clients are partitioned into M groups uniformly at random. Each group then trains the M models in the M rounds in the frame in a round-robin manner.

The error, for a candidate multi-model federated learning algorithm $\mathcal{P}$, for each model is defined as the distance between each model's global weight at round t and its optimizer. Our analytical results show that when the objective function is strongly convex and smooth, for MFA-Rand, an upper bound on the error decays as $\mathcal{O}(1/\sqrt{T})$, and for MFA-RR, the error decays as $\mathcal{O}(1/T)$. The latter result holds when local stochastic gradient descent at the clients transitions to full gradient descent over time. This allows MFA-RR to be considerably faster than FedCluster [5], an algorithm similar to MFA-RR. Further, we study the performance gain in multi-model FL over single-model training under MFA-Rand. We show via theoretical analysis that training multiple models simultaneously can be more efficient than training them sequentially, i.e., training only one model at a time on the same set of clients. Via synthetic simulations we show that MFA-RR typically outperforms MFA-Rand. Intuitively this is because under MFA-RR, each client trains each model in every frame, while this is not necessary in the MFA-Rand. Our data-driven experiments prove that training multiple models simultaneously is advantageous over training them sequentially.

1.2 Related Work

The federated learning framework and the FedAvg algorithm was first introduced in [15]. Convergence of FedAvg has been studies extensively under convexity assumption [12] and non-convex setting [13].

Personalised FL is an area where multiple local models are trained instead of a global model. The purpose of having multiple models [7] is to have the local

models reflect the specific characteristics of their data. Although there are multiple models, the underlying objective is same. This differentiates personalised FL from our setting, which involves multiple *unrelated* models.

Clustered Federated Learning proposed in [16] involved performing FedAvg and bi-partitioning client set in turn until convergence is reached. FedCluster proposed in [5] is similar to the second algorithm we propose in this work. In [5], convergence has been guaranteed in a non-convex setting. However, the advantages of a convex loss function or the effect of data sample size has not been explored in this work. Further, the clusters, here, are fixed throughout the learning process.

Training multiple models in a federated setting has been explored in [10] also. However, the setting and the model training methodology are different from ours [2]. In [10], client distribution among models has been approached as an optimization problem and a heuristic algorithm has been proposed. However, neither an optimal algorithm is provided nor any convergence guarantees.

2 Setting

We consider the setting where M models are trained simultaneously in a federated fashion. The server has global version of each of the M models and the clients have local datasets (possibly heterogeneous) for every model. The total number of clients available for training is denoted by N. We consider full device participation with the set of clients divided equally among the M models in each round. In every round, each client is assigned *exactly one model* to train. Each client receives the current global weights of the model it was assigned and performs E iterations of stochastic gradient descent locally at a fixed learning rate of η_t (which can change across rounds indexed by t) using the local dataset. It then sends the local update to its model, back to the server. The server then takes an average of the received weight updates and uses it to get the global model for the next round. Algorithm 1 details the training process of multi-model FL

Table 1. Common variables used in algorithms

Variable Name	Description
n	round number
M	Number of Models
$\mathcal{S}_T$	Set of all clients
N	Total number of clients $= \lvert\mathcal{S}_T\rvert$
$\mathcal{P}$	Client selection algorithm
trainModel[c]	Model assigned to client c
globalModel[m]	Global weights of model m
localModel[m, c]	Weights of m^{th} model of client c

Algorithm 1. Pseudo-code for M-model training at server

Input: $\mathcal{S}_T$, algorithm $\mathcal{P}$
Initialize: globalModel, localModel
globalModel$[m] \leftarrow \mathbf{0}\ \forall\ m \in \{1, 2, .., M\}$
Initialise parameters relevant to algorithm $\mathcal{P}$ in round 0
repeat
  localModel$[m, c] \leftarrow \mathbf{0}\ \forall\ c \in \mathcal{S}_T, m \in \{1, 2, .., M\}$
  Update parameters relevant to algorithm $\mathcal{P}$
  trainModel $\leftarrow$ call function for $\mathcal{P}$ ▷ MFA_Rand(t) or MFA_RR(t)
  for $c \in \mathcal{S}_T$ **do**
    $m \leftarrow$ trainModel$[c]$
    localModel$[m, c] \leftarrow$ globalModel$[m]$
    localUpdate[m,c] $\leftarrow$ update for localModel$[m, c]$ after E iterations of gradient descent
  end for
  for $m \in \{1, 2, .., M\}$ **do**
    globalUpdate[m] $\leftarrow$ weighted average of localUpdate[m,c]
    globalModel$[m] \leftarrow$ globalModel$[m]$ - globalUpdate[m]
  end for
until $n =$ max iterations

with any client selection algorithm $\mathcal{P}$. Table 1 lists out the variables used in the training process and their purpose.

The global loss function of the m^{th} model is denoted by $F^{(m)}(\mathbf{w})$, the local loss function of the m^{th} model for the k^{th} client is denoted by $F_k^{(m)}(\mathbf{w})$. For simplicity, we assume that, for every model, each client has same number of data samples, $\mathcal{N}$. We therefore have the following global objective function for m^{th} model,

$$F^{(m)}(\mathbf{w}) = \frac{1}{N} \sum_{k=1}^{N} F_k^{(m)}(\mathbf{w}). \tag{1}$$

Additionally, $\Gamma^{(m)} = \min(F^{(m)}) - \frac{1}{N} \sum_{k=1}^{N} \min(F_k^{(m)}) \geq 0$.

We make some standard assumptions on the local loss function, also used in [6,12]. These are,

Assumption 1. *All $F_k^{(m)}$ are L-smooth.*

Assumption 2. *All $F_k^{(m)}$ are μ-strongly convex.*

Assumption 3. *Stochastic gradients are bounded: $\mathbb{E}||\nabla F_k^{(m)}(\boldsymbol{w}, \xi)||^2 \leq (G^{(m)})^2$.*

Assumption 4. *For each client k, $F_k^{(m)}(\boldsymbol{w}) = \frac{1}{\mathcal{N}} \sum_{y=1}^{\mathcal{N}} f_{k,y}^{(m)}(\boldsymbol{w})$, where $f_{k,y}^{(m)}$ is the loss function of the y^{th} data point of the k^{th} client's m^{th} model. Then for*

each $f_{k,y}^{(m)}$,

$$||\nabla f_{k,y}^{(m)}(\boldsymbol{w})||^2 \leq \beta_1^{(m)} + \beta_2^{(m)}||\nabla F_k^{(m)}(\boldsymbol{w})||^2,$$

for some constants $\beta_1^{(m)} \geq 0$ *and* $\beta_2^{(m)} \geq 1$.

The last assumption is standard in the stochastic optimization literature [1,8].

2.1 Performance Metric

The error for a candidate multi-model federated learning algorithm $\mathcal{P}$'s is the distance between the m^{th} model's global weight at round t, denoted by $\overline{\mathbf{w}}_{\mathcal{P},t}^{(m)}$ and the minimizer of $F^{(m)}$, denoted by $\mathbf{w}_*^{(m)}$. Formally,

$$\Delta_{\mathcal{P}}^{(m)}(t) = \mathbb{E}||\overline{\mathbf{w}}_{\mathcal{P},t}^{(m)} - \mathbf{w}_*^{(m)}||. \tag{2}$$

3 Algorithms

We consider two variants of the Multi-FedAvg algorithm proposed in [2]. For convenience we assume that N is an integral multiple of M.

3.1 Multi-FedAvg-Random (MFA-Rand)

The first variant partitions the set of clients $\mathcal{S}_T$ into M equal sized subsets $\{\mathcal{S}_1^t, \mathcal{S}_2^t, ..., \mathcal{S}_M^t\}$ in every round t. The subsets are created uniformly at random independent of all past and future choices. The M subsets are then matched to the M models with the matching chosen uniformly at random.

Algorithm 2 details the sub-process of MFA-Rand invoked when client-model assignment step (step 6) runs in Algorithm 1. An example involving 3 models over 6 rounds has been worked out in Table 2.

Table 2. MFA-Rand over 6 rounds for 3 models.

Round	Model 1	Model 2	Model 3
1	$\mathcal{S}_1^1$	$\mathcal{S}_2^1$	$\mathcal{S}_3^1$
2	$\mathcal{S}_1^2$	$\mathcal{S}_2^2$	$\mathcal{S}_3^2$
3	$\mathcal{S}_1^3$	$\mathcal{S}_2^3$	$\mathcal{S}_3^3$
4	$\mathcal{S}_1^4$	$\mathcal{S}_2^4$	$\mathcal{S}_3^4$
5	$\mathcal{S}_1^5$	$\mathcal{S}_2^5$	$\mathcal{S}_3^5$
6	$\mathcal{S}_1^6$	$\mathcal{S}_2^6$	$\mathcal{S}_3^6$

Algorithm 2. Pseudo-code for MFA-Rand

```
procedure . MFA_RAND(t)
    {S_1^t, S_2^t, ..., S_M^t} ←Partition S_T randomly into M disjoint subsets
    for S ∈ {S_1^t, S_2^t, ..., S_M^t} do
        if S = S_j^t then
            trainModel[c] ← j ∀ c ∈ S
        end if
    end for
    return: trainModel
end procedure
```

3.2 Multi-FedAvg-Round Robin (MFA-RR)

The second variant partitions the set of clients into M equal sized subsets once every M rounds. The subsets are created uniformly at random independent of all past and future choices. We refer to the block of M rounds during which the partitions remains unchanged as a frame. Within each frame, each of the M subsets is mapped to each model exactly once in a round-robin manner. Specifically, let the subsets created at the beginning of frame l be denoted by $\mathcal{S}_j^{(l)}$ for $1 \leq j \leq M$. Then, in the u^{th} round in frame l for $1 \leq u \leq M$, $\mathcal{S}_j^{(l)}$ is matched to model $((j+u-2) \mod M)+1$. Algorithm 3 details the sub-process of MFA-RR invoked when client-model assignment step (step 6) runs in Algorithm 1. An example involving 3 models over 6 rounds has been worked out in Table 3.

Table 3. MFA-RR over 6 rounds (2 frames) for 3 models.

Round	Model 1	Model 2	Model 3
1	$\mathcal{S}_1^1$	$\mathcal{S}_2^1$	$\mathcal{S}_3^1$
2	$\mathcal{S}_3^1$	$\mathcal{S}_1^1$	$\mathcal{S}_2^1$
3	$\mathcal{S}_2^1$	$\mathcal{S}_3^1$	$\mathcal{S}_1^1$
4	$\mathcal{S}_1^2$	$\mathcal{S}_2^2$	$\mathcal{S}_3^2$
5	$\mathcal{S}_3^2$	$\mathcal{S}_1^2$	$\mathcal{S}_2^2$
6	$\mathcal{S}_2^2$	$\mathcal{S}_3^2$	$\mathcal{S}_1^2$

Remark 1. An important difference between the two algorithms is that under MFA-RR, each client-model pair is used exactly once in each frame consisting of M rounds, whereas under MFA-Rand, a specific client-model pair is not matched in M consecutive time-slots with probability $\left(1-\frac{1}{M}\right)^M$.

Algorithm 3. Pseudo-code for MFA-RR

```
procedure MFA_RR(t)
    if t mod M = 1 then
        l = (t−1)/M + 1
        {S_1^l, S_2^l, ..., S_M^l} ← Partition S_T randomly into M disjoint subsets
    end if
    for S ∈ {S_1^l, S_2^l, ..., S_M^l} do
        if S = S_j^l then
            trainModel[c] ← ((j + t − 2) mod M) + 1 ∀ c ∈ S
        end if
    end for
    return: trainModel
end procedure
```

4 Convergence of MFA-Rand and MFA-RR

In this section, we present our analytical results for MFA-Rand and MFA-RR. We have not included the proofs due to space constraints. The proofs of the theorems can be found in the extended version [3].

Theorem 1. *or MFA-Rand, under Assumptions 1, 2, 3, 4 and with* $\eta_t = \frac{\beta}{t+\gamma}$, *where* $\beta > \frac{1}{\mu}$ *and* $\gamma > 4L\beta - 1$, *we have*

$$\Delta^{(m)}_{MFA\text{-}Rand}(t) \leq \frac{\sqrt{\nu}}{\sqrt{t+\gamma}} M \geq 1 \text{ and } E \geq 1,$$

where ν *and related constants are given in Table 4*

Table 4. Values of ν, B, C and $(\sigma_k^{(m)})^2$ as used in Theorem 1

Constant	Value
ν	$\max\left\{\frac{\beta^2(B+C)}{\beta\mu-1}, \mathbb{E}\|\|\overline{\mathbf{w}}^{(m)}_{\text{MFA-Rand},1} - \mathbf{w}_*^{(m)}\|\|^2(1+\gamma)\right\}$
B	$6L\Gamma^{(m)} + (1/N^2)\sum_{k=1}^{N}(\sigma_k^{(m)})^2 + 8(E-1)^2(G^{(m)})^2$
C	$\left(\frac{M-1}{N-1}\right) E^2(G^{(m)})^2$
$(\sigma_k^{(m)})^2$	$4(\beta_1 + \beta_2(G^{(m)})^2)$

The result follows from the convergence of FedAvg with partial device participation in [12]. Note that B and C influence the lower bound on number of iterations, $T_{\text{MFA-Rand}}(\epsilon)$, required to reach a certain accuracy. Increasing the number of models, M, increases C which increases $T_{\text{MFA-Rand}}(\epsilon)$.

Theorem 2. *Consider MFA-RR, under Assumptions 1, 2, 3, 4 and* $\eta_t = \frac{\beta}{1+\lfloor \frac{t}{M} \rfloor + \gamma}$, *where* $\beta > \frac{1}{\mu}$ *and* $\gamma > \beta L - 1$. *Further,* $\mathcal{N}_s(t)$ *is the sample size for SGD iterations at clients in round* t. *If*

$$\frac{\mathcal{N} - \mathcal{N}_s(t)}{\mathcal{N}} \leq \eta_t \left(\frac{V}{2E\sqrt{\beta_1 + \beta_2 G^2}} \right)$$

for some $V \geq 0$, $\beta_1 = \max_m \beta_1^{(m)}$ *and* $\beta_2 = \max_m \beta_2^{(m)}$, *then,*

$$\Delta^{(m)}_{MFA\text{-}RR}(t) \leq \frac{\phi}{\frac{t}{M} + \gamma} \forall \ M \geq 1 \text{ and } E \geq 1,$$

where $\phi = \frac{\beta E G^{(m)}(M-1)}{M} + \max\left\{ \frac{\beta^2(Y+Z+V)}{\beta\mu - 1}, (1+\gamma)\Delta^{(m)}_{MFA\text{-}RR}(1) \right\}$,
$Y = \frac{LG^{(m)}E^2(M-1)}{2M}$ *and* $Z = LG^{(m)}E(E-1)$.

Here, we require that the SGD iterations at clients have data sample converging sub-linearly to the full dataset. For convergence, the only requirement is $\frac{\mathcal{N} - \mathcal{N}_s(t)}{\mathcal{N}}$ to be proportional to the above defined η_t.

We observe that Y, Z and V influence the lower bound on number of iterations, $T_{\text{MFA-RR}}(\epsilon)$, required to reach a certain accuracy. Increasing the number of models, M, increases Y which increases $T_{\text{MFA-RR}}(\epsilon)$.

A special case of Theorem 2 is when we employ full gradient descent instead of SGD at clients. Naturally, in this case, $V = 0$.

Corollary 1. *For MFA-RR with full gradient descent at the clients, under Assumptions 1, 2, 3, 4 and* $\eta_t = \frac{\beta}{1+\lfloor \frac{t}{M} \rfloor + \gamma}$,

$$\Delta^{(m)}_{MFA\text{-}RR}(t) \leq \frac{\phi'}{\frac{t}{M} + \gamma} \forall\, M \geq 1 \text{ and } E \geq 1,$$

where $\phi' = \frac{\beta E G^{(m)}(M-1)}{M} + \max\left\{ \frac{\beta^2(Y+Z)}{\beta\mu - 1}, (1+\gamma)\Delta^{(m)}_{MFA\text{-}RR}(1) \right\}$,
$Y = \frac{LG^{(m)}E^2(M-1)}{2M}$ *and* $Z = LG^{(m)}E(E-1)$.

Remark 2. MFA-RR, when viewed from perspective of one of the M models, is very similar to FedCluster. However, there are some key differences between the analytical results.

First, FedCluster assumes that SGD at client has fixed bounded variance for any sample size (along with Assumption 3). This is different from Assumption 4 of ours. When Assumption 4 is coupled with Assumption 3, we get sample size dependent bound on the variance. A smaller variance is naturally expected for a larger data sample. Therefore, our assumption is less restrictive.

Second, the effect of increasing sample size (or full sample) has not been studied in [5]. We also see the effect of strong convexity on the speed of convergence. The convergence result from [5] is as follows, $\frac{1}{T}\sum_{j=0}^{T-1} \mathbb{E}||\nabla F(\overline{\mathbf{w}}_{jM})||^2 \leq$

$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$. If we apply the strong convexity assumption to this result and use that $\mu||x - x_*|| \leq ||\nabla F(x)||$, we get $\frac{\mu^2}{T}\sum_{j=0}^{T-1}\mathbb{E}||\overline{\mathbf{w}}_{jM} - \mathbf{w}_*||^2 \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$. Applying Cauchy-Schwartz inequality on the LHS we get, $\left(\frac{1}{T}\sum_{j=0}^{T-1}\mathbb{E}||\overline{\mathbf{w}}_{jM} - \mathbf{w}_*||\right)^2 \leq \frac{1}{T}\sum_{j=0}^{T-1}\mathbb{E}||\overline{\mathbf{w}}_{jM} - \mathbf{w}_*||^2$. Finally, we have $\frac{1}{T}\sum_{j=0}^{T-1}\mathbb{E}||\overline{\mathbf{w}}_{jM} - \mathbf{w}_*|| \leq \mathcal{O}\left(\frac{1}{T^{1/4}}\right)$, for any sampling strategy. With an increasing sample size (or full sample size), we can obtain $\mathcal{O}(1/T)$ convergence. This is a significant improvement over the convergence result of FedCluster.

5 Advantage of Multi-Model FL over Single Model FL

We quantify the advantage of Multi-Model FL over single model FL by defining the gain of a candidate multi-model Federated Learning algorithm $\mathcal{P}$ over FedAvg [15], which trains only one model at a time.

Let $T_1(\epsilon)$ be the number of rounds needed by one of the M models using FedAvg to reach an accuracy level (distance of model's current weight from its optimizer) of ϵ. We assume that all M models are of similar complexity. This means we expect that each model reaches the required accuracy in roughly the same number of rounds. Therefore, cumulative number of rounds needed to ensure all M models reach an accuracy level of ϵ using FedAvg is $MT_1(\epsilon)$. Further, let the number of rounds needed to reach an accuracy level of ϵ for all M models under $\mathcal{P}$ be denoted by $T_{\mathcal{P}}(M, \epsilon)$. We define the gain of algorithm $\mathcal{P}$ for a given ϵ as

$$g_{\mathcal{P}}(M, \epsilon) = \frac{MT_1(\epsilon)}{T_{\mathcal{P}}(M, \epsilon)}. \tag{3}$$

Note that FedAvg and $\mathcal{P}$ use the same number of clients in each round, thus the comparison is fair. Further, we use the bounds in Theorem 1 and Theorem 2 as proxies for calculating $T_{\mathcal{P}}(M, \epsilon)$ for MFA-Rand and MFA-RR respectively.

Theorem 3. *When* $\epsilon < \min_m \Delta^{(m)}_{\textit{MFA-Rand}}(1)$ *and* $M \leq \frac{N}{2}$, *the following holds for the gain of* M*-model MFA-Rand over running FedAvg* M *times*

$$g_{\textit{MFA-Rand}}(M, \epsilon) > 1 \,\forall\, M > 1$$
$$\frac{d}{dM} g_{\textit{MFA-Rand}}(M, \epsilon) > 0 \,\forall\, M \geq 1$$

for all $E \geq 2$ *and for* $E = 1$ *when* $N > 1 + \max_m \left\{\frac{6L\Gamma^{(m)}}{(G^{(m)})^2}\right\}$.

We get that gain increases with M upto $M = N/2$, after which we have the $M = N$ case. At $M = N$, each model is trained by only one client, which is too low, especially when N is large.

For $E = 1$, Theorem 3 puts a lower bound on N. However, the $E = 1$ case is rarely used in practice. One of the main advantages of FL is the lower communication cost due to local training. This benefit is not utilised when $E = 1$.

6 Simulations in Strongly Convex Setting

6.1 Simulation Framework

We take inspiration from the framework presented in [12] where a quadratic loss function, which is strongly convex, is minimized in a federated setting. We employ MFA-Rand and MFA-RR algorithms and compare their performance in this strongly convex setting. In addition to that, we also measure the gain of MFA-Rand and MFA-RR over sequentially running FedAvg in this strongly convex setting.
The global loss function is

$$F(w) = \frac{1}{2N}\left(\mathbf{w}^T\mathbf{A}\mathbf{w} - 2\mathbf{b}^T\mathbf{w}\right) + \frac{\mu}{2}||\mathbf{w}||^2, \tag{4}$$

where $N > 1$, $\mathbf{A} \in \mathbf{R}^{(Np+1)\times(Np+1)}$, $\mathbf{w} \in \mathbf{R}^{(Np+1)}$ and $\mu > 0$.

We define local loss function of the k^{th} client as

$$F_k(\mathbf{w}) = \frac{1}{2}\left(\mathbf{w}^T\mathbf{A}_k\mathbf{w} - 2\mathbf{b}_k^T\mathbf{w}\right) + \frac{\mu}{2}||\mathbf{w}||^2, \tag{5}$$

which satisfies our problem statement $F = \frac{1}{N}\sum_{k=1}^{N} F_k$. The structure of $\mathbf{A}_k$ is such that the loss function of each client is a function of only a subset of the weights in the weight vector $\mathbf{w}$. Moreover, this subset of weights is disjoint across clients. This introduces heterogeneity across clients. A detailed explanation of the framework can be found in the full version [3], which we skip due to space constraints.

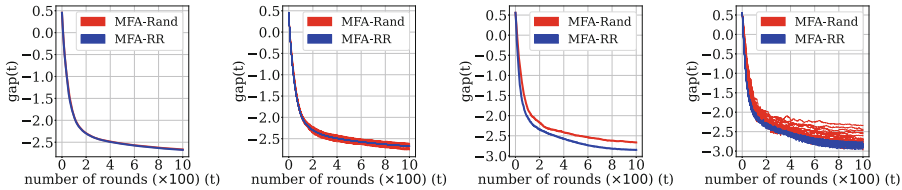
Remark 3. We simulate the minimization of a single global loss function while talking about multi-model learning. The reason behind this is that we are doing these simulations from the perspective of one of the M models. Therefore, M-model MFA-Rand boils down to sampling N/M clients independently every round while M-model MFA-RR remains the same (going over subsets $\{\mathcal{S}_1^l, \mathcal{S}_2^l, .., \mathcal{S}_M^l\}$ in frame l). Further, the gain simulations, here, assume that all M models are of the same complexity.

6.2 Comparison of MFA-Rand and MFA-RR

We consider the scenario where $N = 24$, $p = 4$ and $\mu = 2 \times 10^{-4}$. We take $E = 5$, meaning 5 local SGD iterations at clients. We track the log distance of the current global loss from the global loss minimum, that is

$$gap(t) = \log_{10}(F(w) - F(w_*)) \tag{6}$$

for 1000 rounds. We consider both constant ($\eta_t = 0.1$) and decaying learning rate $\left(\eta_t = \frac{30}{100+t}\right)$ for $M = 2$ and $M = 12$. The constant η_t cases have similar trend as that of decaying learning rate. Due to space constraints, we present



(a) $M = 2$, avg gap (b) $M = 2$, 20 runs (c) $M = 12$, avg gap (d) $M = 12$, 20 runs

Fig. 1. MFA-Rand vs MFA-RR

only the cases of decaying η_t. The results of constant η_t can be found in the full version [3].

We perform 20 sample runs for MFA-Rand and MFA-RR each and compare both algorithms in terms of sample runs and their mean.

In Fig. 1a, we see that the mean performance for MFA-Rand and MFA-RR is similar. However, Fig. 1b reveals that the randomness involved in MFA-Rand is considerably higher than that in MFA-RR, showing the latter to be more reliable.

It is evident from Fig. 1c that MFA-RR, on an average, performs better than MFA-Rand when M is high. Again, Fig. 1d shows that MFA-Rand has considerably higher variance than MFA-RR.

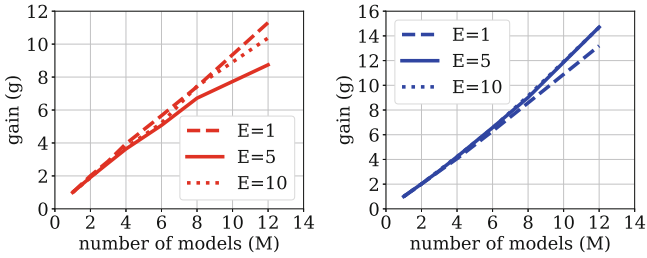
Remark 4. In this set of simulations, each client performs full gradient descent. While the analytical upper bounds on errors suggest an order-wise difference in the performance of MFA-RR and MFA-Rand, we do not observe that significant a difference between the performance of the two algorithms. This is likely because our analysis of MFA-RR exploits the fact that each client performs full gradient descent, while the analysis of MFA-Rand adapted from [12] does not.

6.3 Gain Vs M

We test with $N = 24$ clients for $M \leq 12$ for $E = 1, 5$ and 10. Gain vs M plots in Fig. 2 show that gain increases with M for both MFA-Rand and MFA-RR.

7 Gain Results on Synthetic and Real Federated Datasets

We use Synthetic(1,1) [11] and CelebA [4,14] datasets for these experiments. The learning task in Synthetic(1,1) is multi-class logistic regression classification of feature vectors. Synthetic(1,1)-A involves 60 dimensional feature vectors classified into 5 classes while Synthetic(1,1)-B involves 30 dimensional feature vectors classified into 10 classes. CelebA dataset involves binary classification of face images based on a certain facial attribute, (for example, blond hair, smiling, etc.) using convolutional neural networks (CNNs). The dataset has many options for the facial attribute.



(a) MFA-Rand (b) MFA-RR

Fig. 2. Gain vs M in strongly convex setting

The multi-model FL framework for training multiple unrelated models simultaneously was first introduced in our previous work [2]. We use the same framework for these experiments. We first find the gain vs M trend for Synthetic(1,1)-A, Synthetic(1,1)-B and CelebA. Then, we simulate a real-world scenarios where each of the M models is a different learning task.

7.1 Gain Vs M

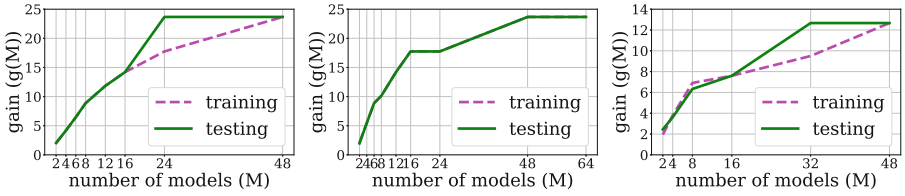
Here, instead of giving M different tasks as the M models, we have all M models as the same underlying task. The framework, however, treats the M models as independent of each other. This ensures that the M models are of equal complexity.

We plot gain vs M for two kinds of scenarios. First, when all clients are being used in a round. Theorem 3 assumes this scenario. We call it full device participation as all clients are being used. Second, when only a sample, of the set of entire clients, is selected to be used in the round (and then distributed among the models). We call this partial device participation as a client has a non-zero probability of being idle during a round.

Full Device Participation: For both Synthetic(1,1)-A and Synthetic(1,1)-B, we have $N = 100$ clients and $T_1 = 70$. For CelebA, we have 96 clients and $T_1 = 75$.

For full device participation, we observe from Fig. 3 that gain increases with M for both Synthetic(1,1) and CelebA datasets with the trend being sub-linear in nature.

Partial Device Participation: For both Synthetic(1,1)-A and Synthetic(1,1)-B, we have $N = 200$ clients (out of which 32 are sampled every round) and $T_1 = 200$. For CelebA, we have 96 clients (out of which 24 are sampled every round) and $T_1 = 75$.



(a) Synthetic(1,1)-A (b) Synthetic(1,1)-B (c) CelebA

Fig. 3. Gain vs M for full device participation

(a) Synthetic(1,1)-A (b) Synthetic(1,1)-B (c) CelebA

Fig. 4. Gain vs M for partial device participation

When there is partial device participation, for both datasets, we observe in Fig. 4 that gain increases with M for the most part while decreasing at some instances. We note that the gain is always more than 1.

Remark 5. It is important to note that the learning task in CelebA dataset invloves CNNs, rendering it a non-convex nature. This, however, does not severely impact the gain, as we still observe it to always increase with M for full device participation.

Remark 6. Although, Theorem 3 assumes full device participation, we see the benefit of multi-model FL when there is partial device participation. For all three datasets, gain is significant and always greater than 1.

7.2 Real-world Scenarios

We perform two types of real world examples, one involving models that are similar in some aspect and the other involving completely different models. In these experiments, T_1 denotes the number of rounds for which single model FedAvg was been run for each model. Further, T_M denotes the number of rounds of multi-model FL, after which each model an accuracy that is at least what was achieved with T_1 rounds of single model FedAvg.

Similar Models: First one tests Theorem 3 where each of the $M(= 9)$ models is a binary image classification based on a unique facial attribute, using CelebA dataset. Table 5 shows the results of our experiment. Based on the values of T_1 and T_M from Table 5, we have the following for training and testing cases.

– Gain in training = $9 \times 50/117 =$ **3.846**
– Gain in testing = $9 \times 50/126 =$ **3.571**.

Table 5. MFA-RR training 9 different CNN models

Facial attribute for classification	**Train Accuracy**		**Test Accuracy**	
	$T_1 = 50$	$T_M = 117$	$T_1 = 50$	$T_M = 126$
Eyeglasses	74.7	84.6	69.1	71.9
Baldness	74.0	74.5	66.8	69.4
Goatee	74.3	83.5	64.7	68.7
Wearing Necklace	73.3	80.3	66.2	72.6
Smiling	72.0	78.7	76.0	79.7
Moustache	74.1	82.1	65.4	72.1
Male	77.1	85.0	58.7	63.5
Wearing Lipstick	75.4	83.8	65.9	72.7
Double Chin	75.3	83.9	63.9	69.3

Heterogeneous Models: In the second one, we do a mixed model experiment where one model is logistic regression (Synthetic(1,1) with 60 dimensional vectors into 5 classes) and the other model is CNN (binary classification of face images based on presence of eyeglasses). Based on T_1 and T_M from Table 6, we get the following values of gain for training and testing cases,

– Gain in training = $2 \times 100/100 =$ **2.0**
– Gain in testing = $2 \times 100/142 =$ **1.41**.

Table 6. MFA-RR training logistic regression and CNN simultaneously

Model Type	**Train Accuracy**		**Test Accuracy**	
	$T_1 = 100$	$T_M = 100$	$T_1 = 100$	$T_M = 142$
Logistic Regression	51.9	52.4	52.8	55.6
Convolutional NN	86.7	87.2	75.8	77.5

8 Conclusions

In this work, we focus on the problem of using Federated Learning to train multiple independent models simultaneously using a shared pool of clients. We propose two variants of the widely studied FedAvg algorithm, in the multi-model setting,

called MFA-Rand and MFA-RR, and show their convergence. In case of MFA-RR, we show that an increasing data sample size (for client side SGD iterations), helps improve the speed of convergence greatly $\left(\mathcal{O}\left(\frac{1}{T}\right)\right.$ instead of $\left.\mathcal{O}\left(\frac{1}{T^{1/4}}\right)\right)$. Further, we propose a performance metric to access the advantage of multi-model FL over single model FL. We characterize conditions under which running MFA-Rand for M models simultaneously is advantageous over running single model FedAvg for each model sequentially. We perform experiments in strongly convex and convex settings to corroborate our analytical results. By running experiments in a non-convex setting, we see the benefits of multi-model FL in deep learning. We also run experiments that are out of the scope of the proposed setting, namely, the partial device participation experiments and the real world scenarios. Here also we see an advantage in training multiple models simultaneously.

References

1. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic programming: an overview. In: Proceedings of the 1995 34th IEEE Conference on Decision and Control, vol. 1, pp. 560–564. IEEE (1995)
2. Bhuyan, N., Moharir, S.: Multi-model federated learning. In: 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS), pp. 779–783. IEEE (2022)
3. Bhuyan, N., Moharir, S., Joshi, G.: Multi-model federated learning with provable guarantees (2022). https://doi.org/10.48550/ARXIV.2207.04330, https://arxiv.org/abs/2207.04330
4. Caldas, S., et al.: Leaf: a benchmark for federated settings. arXiv preprint arXiv:1812.01097 (2018)
5. Chen, C., Chen, Z., Zhou, Y., Kailkhura, B.: FedCluster: boosting the convergence of federated learning via cluster-cycling. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 5017–5026 (2020). https://doi.org/10.1109/BigData50022.2020.9377960
6. Cho, Y.J., Wang, J., Joshi, G.: Client selection in federated learning: convergence analysis and power-of-choice selection strategies. arXiv preprint arXiv:2010.01243 (2020)
7. Eichner, H., Koren, T., McMahan, B., Srebro, N., Talwar, K.: Semi-cyclic stochastic gradient descent. In: International Conference on Machine Learning, pp. 1764–1773. PMLR (2019)
8. Friedlander, M.P., Schmidt, M.: Hybrid deterministic-stochastic methods for data fitting. SIAM J. Sci. Comput. **34**(3), A1380–A1405 (2012)
9. Kairouz, P., et al.: Advances and open problems in federated learning. Found. Trends® Mach. Learn. **14**(1–2), 1–210 (2021)
10. Li, C., Li, C., Zhao, Y., Zhang, B., Li, C.: An efficient multi-model training algorithm for federated learning. In: 2021 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2021). https://doi.org/10.1109/GLOBECOM46510.2021.9685230
11. Li, T., Sanjabi, M., Beirami, A., Smith, V.: Fair resource allocation in federated learning. arXiv preprint arXiv:1905.10497 (2019)

12. Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of FedAvg on Non-IID data. arXiv preprint arXiv:1907.02189 (2019)
13. Li, X., Yang, W., Wang, S., Zhang, Z.: Communication efficient decentralized training with multiple local updates. arXiv preprint arXiv:1910.09126 5 (2019)
14. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3730–3738 (2015)
15. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
16. Sattler, F., Müller, K.R., Samek, W.: Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans. Neural Netw. Learn. Syst. **32**(8), 3710–3722 (2020)
17. Wang, J., Joshi, G.: Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD. In: Talwalkar, A., Smith, V., Zaharia, M. (eds.) Proceedings of Machine Learning and Systems, vol. 1, pp. 212–229 (2019). https://proceedings.mlsys.org/paper/2019/file/c8ffe9a587b126f152ed3d89a146b445-Paper.pdf