FedMDP: A Federated Learning Framework to Handle System and Model Heterogeneity in Resource-Constrained Environments

Ahmed Imteaj,[1,2] **M. Hadi Amini** [3,4]

[1] School of Computing, Southern Illinois University, Carbondale, IL, 62901
[2] Sustainability, Privacy and Edge intElligence for Distributed networks laboratory (SPEED Lab), Carbondale, IL, 62901
[3] Knight Foundation School of Computing and Information Sciences, Florida International University, Miami, FL 33199
[4] Sustainability, Optimization, and Learning for InterDependent networks laboratory (solid lab), FIU, Miami, FL, 33199
ahmed.imteaj@siu.edu, moamini@fiu.edu

Abstract

The advancement of technology, improvement of network infrastructures, and wide availability of internet open up the door of new opportunities to perform on-device inference. Realizing the potential of such advancement, Federated Learning (FL) was invented that facilitates the formation of a powerful model without exposing user data. While successful, it does not consider the combinational case, where the selected FL agents independently craft their local model with heterogeneous architecture and perform computational tasks based on their available resources. In the original FL model, all agents need to agree on a uniform model architecture and are assigned a uniform computational task. However, in a real-life resource-constrained FL setting, agents may not be interested to share their local model architecture details due to privacy and security concerns. Also, the heterogeneous local model architectures cannot be aggregated together on the FL server following the traditional approaches. Moving forward, we may observe straggler agents due to resource-constrained environments, such that any FL agent may find a task as computationally challenging that can prolong the model convergence. To address the above-mentioned challenges regarding agent's local model and resource heterogeneity, we propose an FL framework, FedMDP that can effectively handle federated agents possessing nonidentical local model structure as well as variant local resources using knowledge distillation and dynamic local task allocation techniques. We tested our framework on MNIST and CIFAR100 dataset and observed significant improvement in accuracy in a highly heterogeneous environment. By considering 10 uniquely designed model of the agents, we achieved 15% gain on average compared to the accuracy of the traditional learning methods and observed a few percent lower accuracy compared to the case if the agents' local datasets were pooled and made available for all the network agents.

Introduction

Federated Learning (FL) is a privacy preserving distributed machine learning (ML) scheme that does not require to obtain any user's data in a centralized location, instead, a global predictor model is constructed that is learned through the aggregated knowledge of participating users (1). FL is particularly useful for the applications where the user's data are sensitive and user do not want to share those due to privacy concerns. The FL applications have become widespread that are ranges from healthcare, industrial engineering, to user's input prediction on mobile keyboard.

The state-of-the-art Federated Averaging (FedAvg) (1) algorithm considers an FL server with a participation of n users that jointly collaborate to construct a global model without sharing any potentially privacy sensitive data. Each user generates a local model based on its data and shares the model weights with the FL server. The FL server acts as a central aggregator that aggregates the local model updates of the users and shares the updated global model's weights. Each user learns from the global model and tunes up their local model accordingly. The iterative process between the FL server and the users continues until convergence. On top of preserving privacy, FL provides several other benefits such as autonomy (2), security (3) and efficiency (4) because of its on-device training and distributed decision-making. The authors in (5) discussed the decentralization mechanism of FL and analyzed how it is effective than gossip learning. Despite the benefits, the FL process faces many challenges among which heterogeneity is one the major concerns and appears throughout the learning process. The participated agents in the FL process may have heterogeneous resources in terms of their computational power and bandwidth that was partially solved by the prior works using asynchronous FL technique and further refined through active sampling (6; 7). However, when we have a majority of the agents as stragglers, then asynchronous FL or active sampling failed to work effectively (13). Besides, statistical heterogeneity problem, i.e., nonuniform distribution of data among the agents can also be observed in an FL environment (8; 9). As a result, the agent's local update can be dispersed that may prolong the model convergence. Moreover, the participated FL agents may also possess heterogeneous local models that may cause issues while performing aggregation of the FL server following the state-of-the-art FL process. Some of the FL agents may have simple model while some agents contain complicated and large model architecture. It is because a agent may have sufficient resources to generate a large model while another agent may not be capable to process its local data and generate such a large model. The original FL work assume that the FL agents agree on a particular model architecture and all local models as well

as global model follow that design. However, if we consider that from real-world setting, then any agent may have desire to construct their own unique model. Such situations can be arrived in areas like supply chain, health care, AI services, and finance. For instance, when several department of a supply chain company collaborate without sharing private data, they can craft their own model as per distinct specification. Such supply-chain company would not like to reveal their models because of privacy, and security concerns. Besides, we can consider AI-enabled chat bots that are used for customer service and different companies may have dozens of such service bots. Each service bots may have to deal with different customers and solves variant tasks. It would be beneficial if knowledge of one bot can be shared with others without compromising independency and privacy. Here comes the motivation of FL that preserves privacy though on-device model training and enables each device to learn global knowledge independently. However, one of the core challenges that FL faces is statistical heterogeneity and one unsophisticated way to tackle the statistical heterogeneity is to allow individual model for each agent. Different frameworks, e.g., meta-learning (10), transfer learning (11), and multi-task learning (12) proposed different approaches to handle statistical heterogeneity for non-IID data with acceptable performance; however, their approaches need to perform model customization of the agents up to a certain level. So, instead of centralizing control over agents' models, i.e., customizing agents' models, a mechanism that can enable full model independency can completely tackle the issues of statistical heterogeneity. The authors in (23) proposed an FL model, FedMDR based on a weighted geometric median, which is resilient to the corrupted model injection. The authors in (17; 18; 19; 20) also proposed different FL knowledge distillation technique to handle heterogeneous agent model but none of the approaches are effective while applying in a resource-constrained environment. Another challenge of FL method is systems heterogeneity and it becomes critical when we apply FL in a resource-constrained IoT environment (13). According to the traditional FL method, the task publisher assigns a uniform task to all the selected agents. However, the agents' resources may be limited or heterogeneous and all the selected agents would not be able to perform the assigned task. As a consequence, the FL server may need to wait for a long time for getting update from the straggler agents and it may prolong the overall model convergence. According to the authors of (1; 14), one solution is to drop the straggler agents or not selecting them during model training. For instance, the authors in (22) designed a federated edge learning framework that can select a subset of edge devices as participants by analyzing the downlink channel conditions. However, if the majority of the selected agents are stragglers, then we may have only a few active agents that can significantly reduce the model performance, or some straggler agents may have higher volume of data (15). Beyond model and systems heterogeneity, diverge local model update from the agents is also an issue of federated networks that can be mainly occurred due to false model injection (16). The authors in (21) proposed an FL model, FedMax that can mitigate communication overhead by applying a technique of restricting activation-divergence of the participated agents. However, they did not consider the straggler effects due to resource-constrained agents. The core research question that we try to address in this paper is how we can carry out FL process when the agents have heterogeneous model architectures which is blackbox to others, and the federated networks have a large number of resource-constrained agents that could become stragglers or can infuse false model during global model update.

Contributions

In this paper, we presented our developed an FL framework, FedMDP that can be effectively applied in a highly heterogeneous resource-constrained environment. Our proposed framework can be applied in such a FL setting, where the network agents possess unknown model architectures and the participated agents are resource-constrained devices. We propose to use transfer learning and knowledge distillation to develop a universal framework that enables federated learning when each agent owns not only their private data, but also uniquely designed models and heterogeneous resources. We develop a module that translates knowledge between participants. To this end, We enable variable computational tasks for the participated agents that can significantly mitigate the straggler effects by considering every little computational tasks performed by the agents.

Proposed Approach

Problem Definition

We assume that there are n number of agents in an FL environment. Each agent p owns a small labeled dataset $\mathcal{D}_p := \{(x_i^p, y_i)\}_{i=1}^{N_p}$ and the dataset does not need to be drawn from the similar type of distribution. We considered a large public dataset $\mathcal{D}_0 := \left\{\left(x_i^0, y_i^0\right)\right\}_{i=1}^{N_0}$ that is resided on the server and all FL agents can access that. We enable each user to independently design its own model f_p that is used to perform an assigned computational task. Therefore, each agent may have heterogeneous model architectures. Unlike traditional FL methods,, we aim to preserve agent privacy moving one-step forward by not sharing even the hyper-parameters of the agents. Therefore, a agent model information is not be known to anyone and on top of that, the hyper-parameters of the agent models are not exposed. It would protect the FL process from leaking sensitive information due to void knowledge about agent model architectures. Now, the resource-constrained FL agents may struggle to carry-out learning process if the assigned computational task is too overwhelming. That could result in slow learning process and a majority of such slow learning agents can prolong the model convergence time. We can eliminate that issue by assigning computational tasks among the agents based on their resource-availability. On the whole, the main goal of this paper is to develop an FL model that improves the performance of each agent's model f_p using publicly accessible data $\mathcal{D}_0$ as well as on-device data $\mathcal{D}_p$, and remove the straggler effects by assigning computational tasks based on the agent resources.

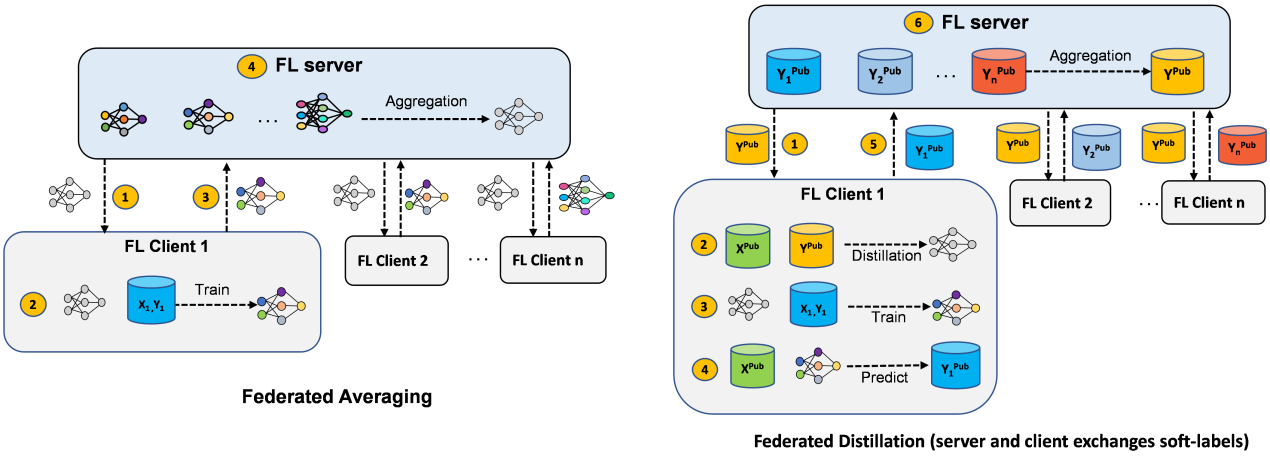


Figure 1: The working process of Federated Averaging and Federated Distillation. In Federated Averaging, training information are transferred between server and agents via model parameters. In Federated Distillation approach, the similar information is shared via soft-label predictions Y^{pub} by the agents on a publicly shared data set X^{pub}.

Proposed Approach

Our proposed FedMDP framework has three phases. In the first phase, we leverage the federated distillation technique by translating the local knowledge. In the second phase, we enable variable local computational tasks based on the agents' resources. Finally, on the third phase, we integrate the distillation technique and dynamic task allocation technique to handle model and system heterogeneity of the network agents.

Leveraging Federated Distillation Technique The key to handle model heterogeneity of the FL agents is communication. In particular, we need to design a translation protocol that enables interpreting the knowledge of a local model that is understandable to the server and to its peers. We consider that each agent has a private dataset with a uniquely designed local model. To carry-out FL process, each agent's local model needs to translate to a standard format. To translate the knowledge of the local models, we propose to employ knowledge distillation technique, where the smaller local models are trained based on the large-sized public dataset residing on the server. That means the translator will be developed using knowledge distillation technique. The central server is mainly responsible for collecting the translated knowledge and spreading a consensus across the FL network. Further, the agents share their output class scores which is evaluated considering the agent's performance on the public dataset.

At first, each participated federated agents generates a model using its local data. Then each agent computes class scores on the public dataset and shares with the server. The server generates an aggregated class scores that are received from all the agents. That aggregated class score is the latest consensus for the public dataset. For the next iteration round, each agent downloads the latest consensus and trains its model based on the public dataset. After that, each agent retrain its local model utilizing its private data and computes an updated class score for the public dataset. Each agent again shares its updated class score and the server generates a latest consensus. The server-agent interaction continues until the consensus class score reach to a target. The working procedure of federated distillation technique is explained below: At the starting of each federated distillation round, the FL server selects a subset of interested agent for the training process and each selected agent can synchronize with the server by downloading the latest soft-labels Y^{Pub} (aggregation of all previously participated agents soft-labels) on public dataset. Each selected agents update their models through model distillation technique. The publicly available dataset and downloaded latest soft-labels are used to generate a distilled model on each agent side. Each agent train the distilled model by applying its own-device local data and generate an updated local model, i.e., each agent learns from the global knowledge. After that, the local model's class scores on the public dataset are generated. Each agent shares the latest soft-label or class score with the FL server. The server performs aggregation on all the collected soft-labels from the agents and computes an aggregated soft-labels Y^{Pub} for the next communication round.

Enabling Partial Computational Tasks In a resource-constrained FL environment, each agent may not be able to perform a given computational task. If we consider the conventional FedAvg (1) algorithm, then an uniform task is assigned to all the selected agents. However, the selected agents may not be homogeneous in terms of their resources (e.g., model architecture, processing power, memory, bandwidth). As a consequence, the majority of the agents would become stragglers during a training process and model convergence would be prolonged. The straggler issue can be observed in the distillation approach, particularly, when a agent applies a distilled model on its private data and generates a new local model (Step 3 in Section). Therefore, it is not reasonable to assign a same computational tasks to all the selected agents. Instead of that, enabling dynamic assignment of the local computational tasks by analyzing the

agent's resources can be effective for a resource-constrained FL environment. Allowing a flexible amount of work helps to solve local objectives inexactly and assists to tune up the number of communication vs. local computations. While too many local epochs can overfit the model, a smaller number of local epochs increases communication overhead as well as convergence time (24). Therefore, it is required to set local epoch through proper tuning to ensure robust convergence. We incorporate a generalization of FedAvg algorithm that entitles the straggler agent to perform partial amount of works instead of the whole task. We can allow partial works for our federated agents that are selected through trust and resource-aware strategy and we can define the ϕ_p^i -inexactness for federated agent p at training round i:

Definition 2 (φ_p^i **-inexact solution**). Let us consider a function $\mathcal{G}_p(w; w_i) = \mathcal{F}_p(w) + \frac{\beta}{2}\|w - w_i\|^2$, and $\varphi \in [0, 1]$, we call w^* is a φ_k^c -inexact solution of $\min_w \mathcal{G}_p(w; w_i)$ if $\|\nabla \mathcal{G}_p(w^*; w_i)\| \leq \varphi_k^i \|\nabla \mathcal{G}_p(w_i; w_i)\|$, where $\nabla \mathcal{G}_p(w; w_i) = \nabla \mathcal{F}_p(w) + \beta(w - w_i)$.

Here, φ_p^i determines how much local computation is needed to perform by the device p in communication round i to solve local problems. That means, φ_p^i is the representation of the variable local iterations of the agents. The systems heterogeneity can be handled by relaxing the φ_p^i -inexactness. In Figure **1**, we presented a conceptual visualization of allowing partial amounts of work to be performed by 3 heterogeneous agents. From the figure, we can see that, agent 1 and n are performing 30% and 48% of the overall tasks due to resource-limitations while the second agent is performing the whole task because of its available resources. If we explain it in more simplified way, then let consider that the task publisher expects 100 local epochs to be performed by all the selected FL agents. However, due to resource-constraint issues, some of the agents may not be able to perform 100 local epoch for generating their local models. For such a case, the weak agents are allowed to perform lower number of local epochs, e.g., the first and forth agents need to perform only 30 and 48 local epoch if the overall computational task is 100 local epochs for that training round.

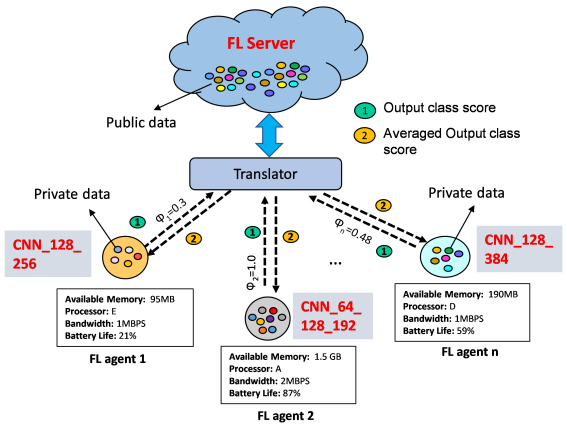


Figure 2: Handling model heterogeneity and systems heterogeneity of federated agents through our proposed approach.

Integrating Federated Distillation and Dynamic Local Task Allocation After leveraging federated distillation for translating local knowledge and enabling dynamic local task allocation to handle straggler agents, we propose an FL framework by integrating both of them which is capable to handle model and system heterogeneity. We graphically represented our proposed *FedMDP* framework in Figure **2**. From that figure, we can see the agents hold variant resources in terms of their available memory, processor, bandwidth and battery life, and possess heterogeneous model architectures (e.g., CNN_128_256, CNN_64_128_192, CNN_128_384, etc.). Each agent performs variable computational task as per their resources (e.g., 0.3, 1.0, 0.48) and generates a local model, After that, each local model compute a class score on the available dataset and share that with the FL server. On the server, we have a translator that aggregates the generated class score that holds the overall feedback of the participated network agents. To reveal more technical insight, we presented the details of our *FedMDP* framework in Algorithm **1**.

Algorithm 1: FedMDP Framework for enabling FL for heterogeneous systems and heterogeneous models. Here, $\mathcal{P}$ denotes the selected agents for FL process.

Input: Public dataset $\mathcal{D}_0$, private datasets $\mathcal{D}_\mathcal{P}$, independently designed model $f_\mathcal{P}, k = 1 \ldots m$.
Output: Trained model $f_\mathcal{P}$.
Registration: Each interested agent, $\mathcal{I}_a$ commits registration.
Initialize set timeout t^i, and select a set of agent, $\mathcal{P}$ for training
Transfer learning: Each $\mathcal{P}$ trains $f_\mathcal{P}$ to convergence on the $\mathcal{D}_0$ and then on $\mathcal{D}_\mathcal{P}$.
for i $= 1, 2 \ldots$ **do**
  Communicate: Each $\mathcal{P}$ computes the class scores $f_\mathcal{P}(x_i^0)$ on the public dataset, and transmits the result to a central server
  Aggregate: The server computes an updated consensus, which is an average $\tilde{f}(x_i^0) = \frac{1}{m}\sum_\mathcal{P} f_\mathcal{P}(x_i^0)$
  Distribute: Each $\mathcal{P}$ downloads the updated consensus $\tilde{f}(x_i^0)$
  Each $\mathcal{P}$ finds a $w_\mathcal{P}^{i+1}$ which is a φ_k^i -inexact minimizer of: $w_\mathcal{P}^{i+1} = F_\mathcal{P}(w) + \frac{\beta}{2}\left\|w - w^i\right\|^2$ and determines maximum feasible number of local epochs, $E_{par}^\mathcal{P}$
  Digest: Each $\mathcal{P}$ trains its model $f_\mathcal{P}$ to approach the consensus $\tilde{f}$ on the public dataset $\mathcal{D}_0$
  Revisit: Each $\mathcal{P}$ trains $f_\mathcal{P}$ on $\mathcal{D}_\mathcal{P}$ for the determined feasible local epochs, $E_{par}^\mathcal{P}$

Here, we consider a public dataset $\mathcal{D}_0$ (which is available to all federated agents), private datasets (which is possessed by each agent and may vary from agent to agent), and

individual agent's model architecture as input (line **1**). The output that we expect for each agent is a trained model $f_{\mathcal{P}}$ that learns from the global knowledge and on-device data (line **2**). Initially, each agent needs to register itself to join in a network (line **3**). After that, training time window, and number of participated agents are initialized (line **4**). Each agent trains its local model $f_{\mathcal{P}}$ to convergence on the public dataset $\mathcal{D}_0$ and then on its private dataset $\mathcal{D}_{\mathcal{P}}$ (line **5**). On each communication round i, each agent generates a score by classifying all the sample of the public dataset $\mathcal{D}_0$ and shares the class score with the FL server (line **6-7**). Upon receiving class scores from all the participated agents, the server performs aggregation on the class scores and generates a consensus that reflects the cumulative knowledge of all the participated agents (line **8**). The latest consensus is shared with all the participated agents and each agent download that updated consensus $\tilde{f}\left(x_i^0\right)$ (line **9**). After that, each participated agent exploits its local solver to determine inexact minimizer $\varphi_{\mathcal{P}}^i$ for solving its objective function, i.e., the number of local epochs that is viable to perform locally in order to evaluate the class score (line **10**). Through on-device model training, each agent approach towards consensus $\tilde{f}$ for public dataset $\mathcal{D}_0$ (line **10**). Finally, each agent $\mathcal{P}$ train its local model $f_{\mathcal{P}}$ on its private dataset $\mathcal{D}_{\mathcal{P}}$ for the determined feasible local epoch, $E_{par}^{\mathcal{P}}$.

Experimental Evaluation

We perform the evaluation of our proposed FL framework in two different settings. In the first test experiment, we consider the MNIST as the public data and a subset of FEMNIST as the private data. We prepare i.i.d. simulation setting by randomly selecting samples from FEMNIST as the private dataset of the agents. For non-i.i.d. case, each participated agent is given only the letters written by a writer (instead of giving handwriting of all writers), and the agent need to classify letters of all writers during testing period. In the second test experiment, we consider CIFAR10 as the public dataset and CIFAR100 as the private dataset, which possesses 100 subclasses with a 20 superclasses, e.g., leopard, tiger, wolf, bear and lion. The main prediction task for the second experiment is to classify image samples into correct subclasses while in the non i.i.d. case, each participated agent holds image samples of one subclass from every superclass, and that agent needs to classify the data that are from other subclassess of every superclass. For instance, an FL agent who has seen leopard during on-device model training is assigned task to classify tiger, wolf, bear, or, lion. The knowledge sharing mechanism of the FL process makes it possible to predict unforeseen events or objects.

In our FL simulation setting, we consider 10 FL participants that has unique convolutional neural networks (CNNs). The CNNs differ in terms of number of layers and number of channels. We considered heterogeneous model architectures (layer, output shape, and number of parameters) of the FL agents, i.e., each agent has uniquely designed model architecture. Now, if we want to aggregate the heterogeneous model architectures, then the traditional FL methods (which performs weighted average of the local model) would fail to generate a global model. To tackle such issues, initially, the participants are trained based on the public dataset until they reach a target convergence. Then, each participant carries-out on-device training on the private dataset. After that, the participants shares the output class scores on the public data and the server generates an aggregated consensus on the labels of the publicly available data. The aggregated consensus is shared with the participants and they tune-up their model accordingly. We evaluated the pre-trained accuracy of the agents considering MNIST and CIFAR100 dataset. For MNIST dataset,we set up three convolutional layer filters and a dropout rate, and observed an accuracy of above 95% for all 10 participated agents (Table **1**). In turn, For CIFAR100 dataset, we set up four convolutional layer filters and a dropout rate, that gives us an accuracy of above 72% for all 10 participated agents (Table **2**). We simulated the model accuracy for MNIST IID dataset by not considering any partial works and we can observe how some of the agents' performance degrades due to straggler effects (see Figure **3(a)**). However, leveraging our proposed Fed-MDP model (which selects effective agents and also allows partial works) generates an effective global model that helps every agent including the stragglers to improve their local model (see Figure **3(b)**). Further, we simulated the model accuracy for MNIST non-IID dataset by again not considering any partial works and we can observe how some of the agents' performance becomes even worse due to straggler effects in non-IID setting (see Figure **4(a)**). However, such performance degradation are effectively handled by our Fed-MDP model which can deal with statistical heterogeneity and can scale up any diverse local model update (see Figure **4(b)**).

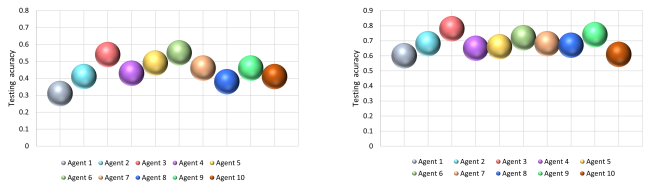
Table 1: Agent pre-trained accuracy for MNIST dataset.

Model	Three Conv. Layer Filters n_1 —n_2 —n_3	Dropout rate	Pre-trained Accuracy
1	128—256 — None	0.2	96.4%
2	128 — 384 — None	0.2	96.6%
3	128 — 512 — None	0.2	96.0%
4	256 — 256 — None	0.3	98.2%
5	256 — 512 — None	0.4	95.3%
6	64 — 128 — 256	0.2	98.3%
7	64 — 128 — 192	0.2	98.4%
8	128 — 192 — 256	0.2	97.9%
9	128 — 128 — 128	0.3	98.9%
10	128 — 128 — 192	0.3	97.4%

However, we realize that assigning an uniform computational task could be overwhelming for any of the participated agents and thus, we infused our dynamic resource allocation strategy (i.e., allowing partial works) for assigning local computational tasks according to the agent's resource availability. To measure the effects of enabling partial works of the agents, we perform a simulation of our federated setting considering system heterogeneity. We assume that, a global clock cycle is remained during the FL process which can specify the time window for executing an assigned computational task. We also assume that, a task publisher will define a global epoch E which needs to be executed by all

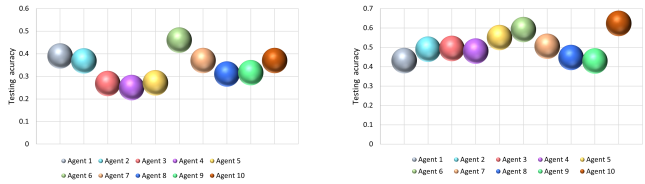
Table 2: Agent pre-trained accuracy for CIFAR100 dataset.

Model	Four Conv. Layer Filters n_1 —n_2 —n_3 — n_4	Dropout rate	Pre-trained Accuracy
1	128 — 256 — None — None	0.2	72.3%
2	128 — 128 — 192 — None	0.2	79.9%
3	64 — 64 — 64 — None	0.2	73.5%
4	128 — 64 — 64 — None	0.3	76.9%
5	64 — 64 — 128 — None	0.4	75.8%
6	64 — 128 — 256 — None	0.2	77.4%
7	64 — 128 — 192 — None	0.2	78.7%
8	128 — 192 — 256 — None	0.2	74.6%
9	128 — 128 — 128 — None	0.3	78.7%
10	64 — 64 — 64 — 64	0.2	75.9%



(a) (b)

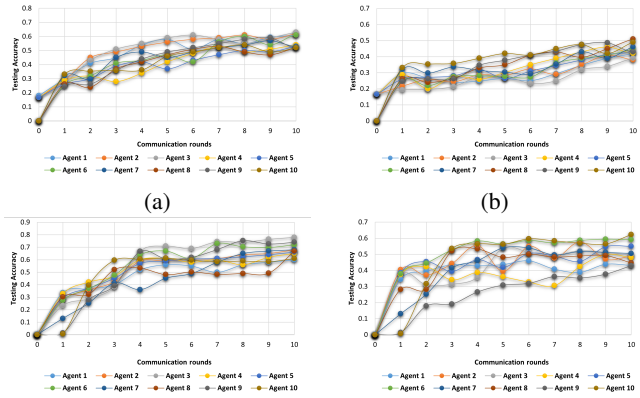
Figure 3: Model accuracy (a) without allowing partial works and (b) enabling partial works (our proposed approach) considering federated agents including stragglers in presence of model heterogeneity and systems heterogeneity using MNIST IID dataset.



(a) (b)

Figure 4: Model accuracy (a) without allowing partial works and (b) enabling partial works (our proposed approach) considering federated agents including stragglers in presence of model heterogeneity and systems heterogeneity using MNIST Non-IID dataset.

resource-sufficient participants. In case, a selected agent k has resource-constraints, then that agent can perform fewer epochs by evaluating the feasible amount of computational works it can perform on communication round i (φ_k^c) using a function consisting of a global cycle and its available resources. We evaluate the learning progress of our considered 10 heterogeneous agents to how those agents collaborate with each other despite having model and system heterogeneity. In Figure **5(a)** and Figure **5(b)**, we present the simulation results of the accuracy of those 10 agents considering both IID and non-IID setting for the MNIST dataset, which depicts that the model quality of each agent is improving as the communication round increases. We also considered CIFAR100 dataset to check the performance of our FedMDP framework. In Figure **5(c)** and Figure **5(d)**, we show the the accuracy of the agents considering both CIFAR100 IID and non-IID dataset, which demonstrate the effectiveness of our proposed FedMDP framework. From the simulation, we can observe that the agents achieve higher accuracy in both of the IID settings as similar type of data are distributed among the agents in such a case. As a result, if an agent fail to perform the whole computational task, other agents that capture the similar knowledge can cover up the loss. In turn, in non-IID settings, we achieve a little less accuracy than IID setting which is obvious, still the agents achieve a notable accuracy within a very few communication round despite having high heterogeneity in their local data.



(a) (b)

(c) (d)

Figure 5: Agent's learning progress in different communication rounds in spite of systems and model heterogeneity through transfer learning-based knowledge distillation and leveraging partial computational tasks on (a) CIFAR100 IID dataset, (b) CIFAR100 Non-IID dataset, (c) MNIST IID dataset and (d) MNIST Non-IID dataset.

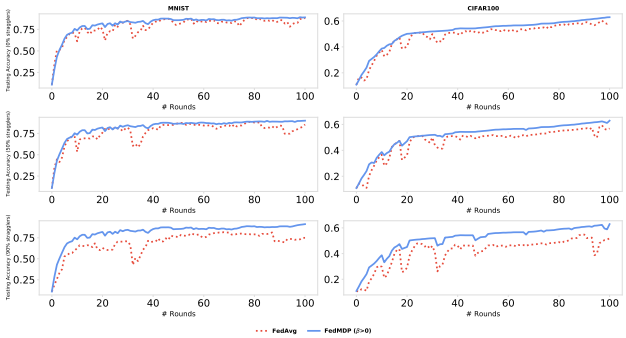


Figure 6: Testing accuracy of FedAvg model and our proposed model in presence of different percentages of stragglers (i.e., 0%, 50%, and 90%)

To evaluate the performance of our proposed model in presence of straggler effects, we defined various number of agents that could be stragglers, e.g., $0\%, 10\%, 20\%, 50\%, 90\%$, where 0% straggler indicates that all the participated agents are able to perform the assigned computational tasks (i.e., global epoch E), and

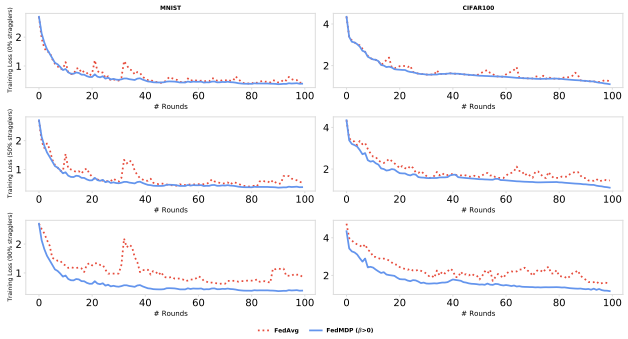


Figure 7: Training loss of FedAvg model and our proposed model in presence of different percentages of stragglers (i.e., 0%, 50%, and 90%)

90% agents are stragglers means only 10% agents could perform the whole task. In such a situation, the state-of-the-art FedAvg algorithm simply drops the stragglers from the training round which could prolong the convergence time, or the global model could never reach to the target convergence of there is a high number of stragglers. Instead of that, if we could count every little computational tasks that are performed by the resource-constrained agents, then we would mitigate straggler effects and also the model convergence would be accelerated. In Figure **6**, we demonstrated the validation of our hypothesis by simulating the training loss considering different number of stragglers $(0\%, 50\%, 90\%)$ and we achieved higher training loss of the global model compare to the FedAvg method. We also simulated the testing accuracy of FedMDP and FedAvg, and it is clear that the FedMDP outperforms the FedAvg, particularly when a high number of network agents are stragglers (see Figure **7**). From these simulations, it is clear that heterogeneity has a negative impact on the agent's learning process and dropping the straggler agent can significantly degrade the model performance. Instead, leveraging our proposed FedMDP model handles model heterogeneity through knowledge distillation technique and also helps to achieve higher model accuracy by considering every little computational tasks performed by the agents.

We also investigate our proposed FedMDP model by considering two different settings. In the first inspection, we limit the agent's local computational task as exactly 1 epoch, i.e., each selected agent needs to perform only a single local iteration on its local data using its local resources and learning from global model. In that investigation, we found that our proposed FedMDP model performs better than the FedAvg model. In our next investigation, we checked the performance of FedMDP and FedAvg on a synthetic Independent and Identically Distributed Data (IID) dataset that does not have any statistical heterogeneity across different network agents. For such a dataset, the FedAvg method performs better because it simply drops the stragglers, and consider the contributions of other agents that already holds the similar type of knowledge. However, in non-IID dataset, if we simply drop the stragglers, then we may lose valuable information that are possessed by that agent. That is why, in non-IID dataset, our proposed method obtains Superior performance than the FedAvg method.

We analyze several valuable aspects of our simulation results:

1. We evaluate the accuracy an agent could have achieved considering the same simulations settings with the presence of straggler agents. All local data of the agents are pooled together and shared with all the agents (see Table **V**). Our proposed model accelerates the performance of all participated agents only a few percent less than the performance of the pooled data in presence of stragglers.
2. Our proposed model can handle extreme level of model heterogeneity. We considered several models that have low prediction performance, i.e., fully connected neural networks with two layers. If such models contribute equally as the advanced models, then the overall model performance is hindered. If we suppress the contribution of the resource-limited agents that can not perform the whole computational tasks and possess a low-quality model, then our model performs better.

Conclusion

In this paper, we proposed an FL framework, *FedMDP* that can deal with agents with heterogeneous local model architectures as well as heterogeneous locally available resources. Through the generalization of FedAvg algorithm and knowledge translation of the federated agents, our developed framework can accelerate the model convergence and upgrade the knowledge of the uniquely designed models of the agents. We tested our framework on various datasets and tasks and we demonstrated the effectiveness of FedMDP even in a highly resource-constrained environment. Our simulations results prove that the infusion of knowledge distillation and allowing partial works significantly improve the model quality of the network agents and also cut-off the negative impacts due to slow agents towards model convergence. In future, we will investigate more sophisticated privacy mechanism while sharing class score in the distillation process and will explore the optimal hyper-parameter tuning for our proposed framework.

Acknowledgement

This material is partly based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STCIN00001-05-00. This material is also based upon work partly supported by the U.S. Department of Homeland Security under Grant Award Number, 2017-ST-062-000002. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] McMahan, Brendan, 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.

[2] Niknam, Solmaz, Harpreet S. Dhillon, and Jeffrey H. Reed. "Federated learning for wireless communications: Motivation, opportunities, and challenges." IEEE Communications Magazine 58, no. 6 (2020): 46-51.

[3] Ma, Chuan, Jun Li, Ming Ding, Howard H. Yang, Feng Shu, Tony QS Quek, and H. Vincent Poor. "On safeguarding privacy and security in the framework of federated learning." IEEE network 34, no. 4 (2020): 242-248.

[4] Sattler, Felix, Thomas Wiegand, and Wojciech Samek. "Trends and advancements in deep neural network communication." arXiv preprint arXiv:2003.03320 (2020).

[5] Hegedűs, István, Gábor Danner, and Márk Jelasity. "Decentralized learning works: An empirical comparison of gossip learning and federated learning." Journal of Parallel and Distributed Computing 148 (2021): 109-124.

[6] Nishio, Takayuki, and Ryo Yonetani. "Client selection for federated learning with heterogeneous resources in mobile edge." In ICC 2019-2019 IEEE International Conference on Communications (ICC), pp. 1-7. IEEE, 2019.

[7] Lim, Wei Yang Bryan, et al. "Federated learning in mobile edge networks: A comprehensive survey." IEEE Communications Surveys & Tutorials 22, no. 3 (2020): 2031-2063.

[8] Zhang, Chen, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. "A survey on federated learning." Knowledge-Based Systems 216 (2021): 106775.

[9] Sery, Tomer, Nir Shlezinger, Kobi Cohen, and Yonina C. Eldar. "Over-the-air federated learning from heterogeneous data." IEEE Transactions on Signal Processing (2021).

[10] Khodak, Mikhail, Maria-Florina F. Balcan, and Ameet S. Talwalkar. "Adaptive Gradient-Based Meta-Learning Methods." Advances in Neural Information Processing Systems 32 (2019): 5917-5928.

[11] Hangyu Zhu, Jinjin Xu, Shiqing Liu, Yaochu Jin,. "Federated learning on non-IID data: A survey." Neurocomputing, Volume 465, Pages 371-390, ISSN 0925-2312, 2021.

[12] Smith, Virginia, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. "Federated multi-task learning." In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[13] A. Imteaj, U. Thakker, S. Wang, J. Li and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," in IEEE Internet of Things Journal, vol. 9, no. 1, pp. 1-24, 1 Jan.1, 2022, doi: 10.1109/JIOT.2021.3095077.

[14] Bonawitz, Keith, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon et al. "Towards federated learning at scale: System design." Proceedings of the 2nd SysML Conference, Palo Alto, CA, USA, 2019.

[15] Li, Tian, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated optimization in heterogeneous networks." Proceedings of Machine Learning and Systems 2 (2020): 429-450.

[16] Kang, Jiawen, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. "Reliable federated learning for mobile networks." IEEE Wireless Communications 27, no. 2 (2020): 72-80.

[17] Jeong, Eunjeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data." arXiv preprint arXiv:1811.11479 (2018).

[18] Ahn, Jin-Hyun, Osvaldo Simeone, and Joonhyuk Kang. "Wireless federated distillation for distributed edge learning with heterogeneous data." In 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1-6. IEEE, 2019.

[19] Li, Daliang, and Junpu Wang. "Fedmd: Heterogenous federated learning via model distillation." arXiv preprint arXiv:1910.03581 (2019).

[20] Lin, Tao, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. "Ensemble distillation for robust model fusion in federated learning." In Proceedings of 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada. .

[21] Chen, Wei, Kartikeya Bhardwaj, and Radu Marculescu. "Fedmax: mitigating activation divergence for accurate and communication-efficient federated learning." arXiv preprint arXiv:2004.03657 (2020).

[22] Amiri, Mohammad Mohammadi, Sanjeev R. Kulkarni, and H. Vincent Poor. "Federated learning with downlink device selection." In 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 306-310. IEEE, 2021.

[23] Mi, Yuxi, et al. "FedMDR: Federated Model Distillation with Robust Aggregation." In Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data, pp. 18-32. Springer, Cham, 2021.

[24] A. Imteaj and M. Hadi Amini, "FedAR: Activity and Resource-Aware Federated Learning Model for Distributed Mobile Robots," 19th IEEE International Conference on Machine Learning and Applications (ICMLA), 2020, pp. 1153-1160.