Quality-Oriented Federated Learning on the Fly

Fei Wang, Baochun Li, and Bo Li

Abstract

With federated learning, a large number of edge devices are engaged to train a global model collaboratively using their local private data. To train a high-quality global model, however, recent studies recognized that the quality of model contributions from local training on different edge devices are substantially different. Existing mechanisms for quantifying such model quality are intuitively based on the training loss or model parameters, and fail to capture the effect of highly variable data and heterogeneous resources available on participating edge devices. In this article, we propose a new aggregation mechanism that uses deep reinforcement learning to dynamically evaluate the quality of model updates, with accommodations for both data and device heterogeneity as the training process progresses. By dynamically mapping the quality of local models to their importance during model aggregation, the global training process is able to converge toward the direction of better effectiveness and generalization. We show that our proposed mechanism outperforms its state-of-the-art counterparts, achieving faster convergence and more stable learning progress. Further, the LSTM-TD3 architecture and state representation design in our mechanism allows it to adapt to various unseen federated learning environments with an arbitrary number of local updates.

Introduction

Since its debut, federated learning [1] has emerged as a new paradigm for training machine learning (ML) models in a collaborative and privacy-preserving manner. With federated learning, smart devices, referred to as clients, train local models with their own data. The central server, on the other hand, maintains a consolidated global model by aggregating the local model parameters without transferring sensitive raw data between clients and the server. Despite recent applications in practice [2, 3], federated learning still faces several important challenges with current-generation communication environments. In particular, network conditions and computation resources (such as GPU availability) may vary substantially across the clients, and training data is likely to be non-IID (independently and identically distributed). In these scenarios, different devices will produce model updates with varying levels of qualities, which affect the training performance on the server.

On the other hand, it may be quite challenging to quantitatively determine the quality of each model update. In an extreme case, for example, if a client has a large number of data samples and a more powerful GPU to train on, chances are that it may produce a high quality model with its local data. In contrast, the contributions of model updates may be quite low if a client's data samples are distant from the ones drawn from the overall distributions across all participating clients.

Existing mechanisms have been proposed in the literature to alleviate the performance degradation caused by system heterogeneity, and to generate a global model with higher quality. In general, they sought to design a proper function to estimate the quality of each local update based on certain metrics during each round of training, such as values derived from the training loss or model parameters. These quality metrics are expected to reflect either how well the corresponding client learns on its own data [4], or how close the local model is to the global model or other local model [5, 6, 7]. These metrics are derived from very limited information from the clients, due to the needs to preserving data privacy and minimizing communication costs. In fact, the server is not able to access explicit feedback on how well the training has been going after each round. As a result, metrics used by existing mechanisms may not be sufficient for the global model to converge toward high quality.

Equipped with sufficient resources including both computing and storage capacities, modern edge devices are more willing to participate in a federated learning session, whereas the server has the capacity to handle a limited number of edge devices as its clients. In this context, model updates from the clients offer a wide variety of qualities, and such qualities of model updates will have a significant impact on the aggregated global model during the training process. To speed up convergence, it becomes increasingly important to judiciously reallocate the aggregation weights based on the qualities of model updates. Federated averaging (FedAvg) [1], which assigns weights to models from the clients based only on the number of data samples, is clearly oblivious to the qualities of model updates.

In this article, our ultimate objective is to propose a more general mechanism that evaluates model qualities with accommodations for time-varying network conditions and heterogeneous client capabilities throughout a long-running training process. In other words, our overarching goal is to answer an important question: Does there exist an effective method to quantitatively evaluate the qualities of model updates from the

Digital Object Identifier:
10.1109/MNET.001.2200235

Fei Wang and Baochun Li are with the University of Toronto, Canada; Bo Li is with Hong Kong University of Science and Technology.

clients, leading to speedy convergence during the training process?

A highlight of our original contributions is the design and implementation of Fei,[1] a new federated learning mechanism with a novel aggregation strategy that assigns weights to the local updates according to their qualities. To cope with the time-varying network conditions and heterogeneous device capabilities in practice, we leverage deep reinforcement learning (DRL) to dynamically evaluate the quality of model updates, with accommodations for data and device heterogeneity as the training process progresses. Specifically, we carefully design the state representation with critical metrics inspired by existing designs for the DRL agent to "sense" the clients' learning performance instantly after each round of local training, and to produce an evaluation of update qualities based on model weights and little extra information about the clients. We cast model aggregation as a control problem, and apply our quality evaluation on the weighting in the global aggregation process. With the adoption of the Long-Short-Term-Memory-based Twin Delayed Deep Deterministic Policy Gradient (LSTM-TD3) algorithm, our DRL agent is capable of dealing with an arbitrary number of client updates, and accommodates dynamically changing heterogeneity across the clients.

We implemented our proposed DRL agent on Plato, a new scalable federated learning research framework,[2] and evaluated its performance on several federated learning tasks using FashionMNIST and CIFAR-10 datasets. Our experimental results have shown that Fei can effectively reduce the elapsed wall-clock time when training a global model of high quality and maintain a stable learning progress in different settings. Furthermore, our agent is both flexible when accommodating a varied number of client updates and generalizable to different federated learning datasets or models.

Related Work

Considering the widely diverging qualities of model updates from participating clients, existing studies have proposed new adaptive client selection or alternative aggregation algorithms to improve the ultimate training precision and convergence speed of federated learning. Some existing works, for example, investigated how local model contributions can be quantitatively evaluated with network resource consumption (e.g., reducing communication rounds) [8] or client reputation [9]. Various metrics have been taken into consideration by the existing works to evaluate how likely a client is to be selected or how much weight should be assigned to the gradient updates of a selected client. The way of characterizing the quality of model updates defined in the existing works into the following three categories:

- *Within each local update*. The AFL framework [4] selected an optimized subset of clients, each contributing a model update with higher quality, which is determined by the training loss of the corresponding local model.
- *Across local updates*. Based on the intuition that clients are not independent and not equivalent, FedGP [7] actively selected clients considering the correlation between the loss changes of all the local models using a Gaussian Process.

Considering the widely diverging qualities of model updates from participating clients, existing studies have proposed new adaptive client selection or alternative aggregation algorithms to improve the ultimate training precision and convergence speed of federated learning.

- *Between local and global updates*. To obtain the aggregation weights that represent the quality of each local model, FedAtt [5] and FedDA [10] used the attention mechanism and applied it to the layer-wise parameters to capture the distance between the local and global models. FedAdp [6] assign the aggregation weights based on the node contribution which is measured by the angle between the global gradient and the corresponding local gradient.

Although existing works on characterizing the quality of model updates in federated learning have shown to be effective, they have not been able to accommodate the time-varying nature of client heterogeneity as the training progresses (sometimes over several days). It has also been a lack of in-depth understanding of what quality metrics are the most effective in characterizing the qualities of model updates. In this article, with the objective of further improving the training performance in federated learning, we propose to take the effectiveness of various quality metrics into account and consider the time-varying nature of such model qualities during the training process.

Motivation and Challenges

Getting away from the commonly used assumptions for federated learning, we notice two unique characteristics of a more realistic federated learning environment. They motivate us to utilize deep reinforcement learning to evaluate model update quality. First, it is dynamic over time: clients may opt in and opt out with changing computation or communication capacity during a federated learning session. Reinforcement learning algorithms can generate a policy that facilitates the server to make decisions to adapt to the dynamic changes throughout the entire FL training process. Furthermore, there does not exist any explicit instant feedback at an arbitrary round to inform the server of how good a current local update is to the developing global model, which is also affected by other updates at the same time. Reinforcement learning algorithms can train on those limited observations through repeated federated learning sessions to learn how they can affect the evolution of the global model. Second, it is also diverse: rather than collaboratively training only one machine learning model, a large number of clients may be asked to participate in the federated training of a series of machine learning models on their local data during the allocated time. The machine learning models can have different NN architectures or can be trained on other datasets. An ideal DRL policy can generalize well to unseen environments after it has been deployed.

To date, many researchers have explored the improvement of federated learning using reinforcement learning in several situations. Favor [11] employed a well-designed DRL agent to learn which subset of clients to be selected for local training in each communication round. Zhan

[1] Fei is not an acronym, but if one is preferred, it may stand for feasible and effective idea.

[2] Plato is open-source and available at https://github.com/TL-System/plato, and Fei is available at /tree/main/examples/fei

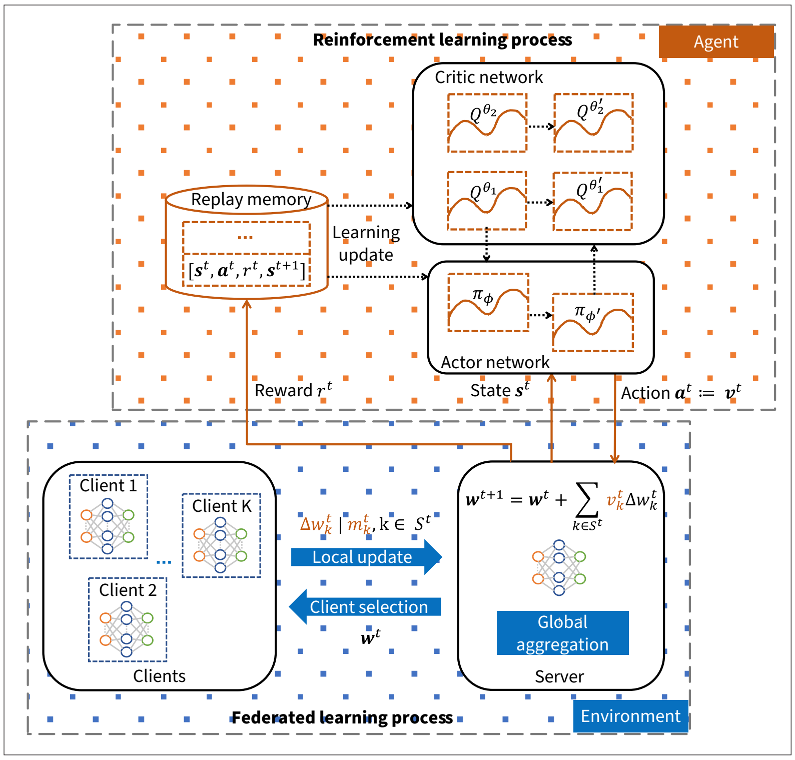

FIGURE 1. Applying deep reinforcement learning to update quality evaluation in global aggregation of federated learning.

et al. [12] designed an incentive mechanism by applying DRL to the aggregator's pricing strategy. However, there are still several critical challenges that have not yet been considered by existing works, which a DRL agent may very likely to encounter in federated learning environments.

Handling a Varied Number of Local Updates

In real-world networks, selected clients may quit the local training job or disconnect with the server at an arbitrary round during a federated learning session, which results in a varied number of updates received at the server at different rounds. An issue occurs that the fixed-sized inputs required by standard neural networks in a DRL agent may not accommodate the arbitrary number of client updates if the state and action space are correlated to each client. As a solution, we take advantage of the flexibility of the recurrent neural networks with the padding technique. In this way, our DRL agent can also be adapted to another federated learning scenario with a different scale of clients without retraining.

Generalization vs. Extra Computation and Communication Overhead

To further make the DRL agent in FL feasible in practice, we need to consider how a model can be generalized beyond the environment it was trained in. It's not worth it to train a DRL model first which introduces much extra computation and communication overhead if it can't also be deployed in other federated learning scenarios (e.g., on another image classification dataset or with another CNN model). Intuitively, those different federated learning scenarios are similar in some sense since what they all do is send and receive neural network models at each communication round between multiple clients and the server. To capture it, we need a DRL formulation that is not coupled with any specific federated learning task, which means the state representation should fall into a similar distribution even though the federated learning environment changes. Besides, to mitigate the computation and communication overhead induced in the DRL pre-training process, the agent is expected to learn rapidly with a design of state and reward that requires limited information from the server and participating clients.

Design of the DRL Agent for Quality Evaluation and Global Aggregation

We are now ready to formulate the global model aggregation process in federated learning as a deep reinforcement learning (DRL) control problem, which is illustrated in Fig. 1. We then explicate how we enhance the flexibility and generalizability of our DRL agent by utilizing recurrent neural networks with the Twin Delayed Deep Deterministic Policy Gradients (TD3) algorithm [13] and multi-task training.

DRL Formulation

Consider a federated learning task where K client devices participate in the training. A DRL agent is embedded in the central server and serves as an evaluator for the quality of updates from the selected clients, based on which the server adaptively generates a new global model using aggregation. The agent takes one action in every time step t, which corresponds to one communication round in the federated learning session. The federated learning session constitutes an evolving environment with which the agent can interact with in each time step t, obtaining observations and rewards from both the server and the selected $\mathcal{S}^t$ clients, and making decisions on the global aggregation weights. The design of the DRL formulation is presented in more detail as follows.

State: The agent observes the interaction between the selected $\mathcal{S}^t$ clients and the server, and measures the quality of each local model update. The state will affect how the agent establishes a sense of how important each local model update is to the global model. In this control problem, we have concerns not only about the size of the state space that affects the convergence of DRL training, but also about the individual privacy of clients. Accordingly, the state space should be limited in terms of the size and the private information contained in it.

Considering both perspectives of individual model updates and the global aggregated model, we propose a new design of the state in time step t, denoted as a vector $\mathbf{s}^t = [s_1^t, s_2^t, \dots, s_k^t, \dots, s_{|\mathcal{S}^t|}^t]$. Each element s_k^t represents a vector of metrics for the corresponding client k, which consists of the following features:

- D_k^t: the number of data samples used for client k's local training in round t;
- τ_k^t: the time, in seconds, taken by local training at client k in round t;
- $\texttt{loss}(\mathbf{w}_k^t; \mathrm{D}_k^t, \mathrm{E})$: the training loss of client k's local model on D_k^t data samples averaged, over E epochs between the last step $t-1$ and the current step t;
- $\texttt{corr}(\nabla \mathrm{F}_k(\mathbf{w}^t), \nabla \mathrm{F}(\mathbf{w}^t))$: the correlation

between the local gradient $\nabla F_k(\mathbf{w}^t)$ of client k and the global gradient $\nabla F(\mathbf{w}^t)$ in round t.

The first feature we choose is D_k^t, which is a primary factor for local training as we mentioned earlier. The local training time τ_k^t informs the agent of the client speed in the asynchronous training setting, which can be affected by different computation capabilities. The average local training loss $\mathtt{loss}(\mathbf{w}_k^t; D_k^t, E)$ is an explicit signal of the model accuracy on its own data. Even with the use of the first three metrics to reflect individual properties of local updates, the contribution of each update to the aggregated global model cannot be decided without taking into account the relation between the local and global model. Given that local models can diverge from the global one especially when the data distribution is non-IID, we further incorporate the correlation

$$\mathtt{corr}\left(\nabla F_k(\mathbf{w}^t), \nabla F(\mathbf{w}^t)\right) = \frac{\langle \nabla F_k(w^t), \nabla F(w^t)\rangle}{\|\nabla F_k(w^t)\| \|\nabla F(w^t)\|}$$

to reveal how close each local model is to the current global one. The local stochastic gradient is calculated by $\nabla F_k(\mathbf{w}^t) = -\Delta \mathbf{w}_k^t/\eta$, where η is the assigned learning rate and then the global gradient is given by

$$\nabla F(\mathbf{w}^t) = \sum_{k=1}^{|S^t|} \left(\frac{D_k^t}{\sum_{k=1}^{|S^t|} D_{k'}^t} \right) \nabla F_k(\mathbf{w}^t)$$

(also used in FedAdp [6]).

With such a design, each selected client k in round t only needs to send the server a minimum amount of extra information m_k^t, that is, D_k^t, τ_k^t, and $\mathtt{loss}(\mathbf{w}_k^t; D_k^t, E)$. The correlation metrics, $\mathtt{corr}(\nabla F_k(\mathbf{w}^t), \nabla F(\mathbf{w}^t))$, are only scalar numbers computed from the model updates and learning rates. As such, our design of the state space will only need to incur a very limited amount of extra computation and communication costs, and will not incur any additional risks of privacy leakage beyond what has already been shared by conventional FL sessions.

As a final step of constructing the agent's state, feature normalization is further used on the four features across the clients, such that all the values per feature fall into the same distribution. In this way, the agent is more capable of interpreting the observations with the presence of different FL environments, and more generalizable as a result.

Action: As we cast the global aggregation with quantification of the update quality as our DRL control problem, the action of step t can be represented as $\mathbf{a}^t = (v_1^t, v_2^t, ..., v_k^t, ..., v_{|S^t|}^t)$, where v_k^t represents the quality value of the corresponding model update from k client. The action space is restricted to $|S^t|$, the number of clients selected per round, instead of K, the number of clients in total.

We apply the quality evaluation to the global aggregation weighting. We normalize the agent's action $\mathbf{a}^t$ to make sure their values sum to 1, and then use the normalized action as the weights of local updates from the corresponding clients when aggregating them into the new global model.

Episode and Reward: In reinforcement learning, episodic tasks have a terminal state. After reaching such a terminal state, the learning will restart in the next episode from an initial state. For the optimization of global aggregation here, one may intuitively regard achieving a target global accuracy as a terminal state of an episode, since that coincides to producing a global model of high quality. However, a well designed episode, accompanied by the design of the reward, is essential for the agent to generalize over different environments, especially considering that the achievable global accuracy is heavily dependent on the specific FL session. Specifically, FL sessions with different machine learning tasks will inherently have different learning complexity, where the progress of global model accuracy can vary significantly. Suppose a DRL agent is learning on a different FL task (e.g., different network architectures, different datasets, or different levels of non-IID data distribution) every a few episodes, then easier machine learning tasks will result in higher accuracies in fewer time steps. In this sense, episodes running these easier tasks will be biased during training.

In reinforcement learning, episodic tasks have a terminal state. After reaching such a terminal state, the learning will restart in the next episode from an initial state.

To make an episode end at an appropriate state, we are concerned about the history of achieved accuracies near the current step t, denoted as acc_hist^t, which is a list of the global model accuracy values obtained using the server-side test dataset in the most recent three time steps. When the standard deviation of the history of accuracies std(acc_hist^t) goes below a certain threshold σ, which suggests that a *steady* global model stage has been reached, the current DRL training episode will terminate, and a new episode (i.e., a new session of FL training) will start. Such a design makes it unnecessary to repeat the entire FL training in every episode.

A reward function should be designed to encourage the desired behavior. The DRL agent is expected to learn to decide the contributions of each local update to the aggregated model and help generate a global model of high quality — one that can achieve high accuracy. However, since the server-side test accuracy is highly correlated to the quality of the global model trained so far, it is not capable of discriminating how effective the taken action is at the previous step. In addition, the global model is expected to progress after aggregating the local training updates, even without any involvement from a DRL agent. The test accuracy of the global model at the current step will be an implicit metric for evaluating how the DRL agent has learned about the quality evaluation of updates.

Instead, we impose a penalty to the agent at every single time step before the end of an episode, and only offer a reward related to the latest history of test accuracies at the end of the episode. The reward function can be expressed as follows:

$$r^t = \begin{cases} \ln \dfrac{\text{EWMA}(\text{acc}_{\text{hist}}{}^t)}{1 - \text{EWMA}(\text{acc}_{\text{hist}}{}^t)} \cdot \beta & \text{if } t \text{ is the last time step,} \\ -1 & \text{otherwise,} \end{cases}$$

where the value EWMA(acc_hist^t) $\in (0, 1)$ represents the exponentially weighted moving average (EWMA) of the global model accuracies stored in the history, whose smoothing parameter

A reward function should be designed to encourage the desired behavior. The DRL agent is expected to learn to decide the contributions of each local update to the aggregated model and help generate a global model of high quality — one that can achieve high accuracy.

is set to 0.9. In this way, not only the accuracy at the end but all accuracies in the history are considered, while larger weights are assigned to accuracies at more recent time steps. We adopt the logistic transformation so that small differences in acc near 1 (e.g., between 0.98 and 0.99) will have a larger differences on the resulted reward. In this way, a higher cumulative reward will be returned to the agent when a global model of higher quality is obtained within fewer communication rounds. The coefficient β decides the importance of the ultimate model quality over the training episode length. For instance, with a larger β, the agent will care more about improving the accuracy than the total consumed time.

Training with TD3

We choose a state-of-the-art reinforcement learning algorithm, Twin Delayed Deep Deterministic Policy Gradients (TD3) [13], to train our agent. As an extension of the Deep deterministic Policy Gradient (DDPG) algorithm which is widely used to solve continuous control problems, TD3 addresses the overestimation of Q values and is therefore more stable and robust to hyperparameters. These characteristics are essential for our agent to have productive learning experiences in a federated learning environment, where time steps can be long due to potentially heavy local training workload.

The structure of neural networks in TD3 is also depicted in Fig. 1. There is a pair of critic networks Q^{θ_1}, Q^{θ_2} for Q-value approximation, and the smaller Q-value between them will be taken for the policy optimization for less bias. TD3 updates the actor and target critic networks every two time steps in a delayed manner, allowing the policy to develop more stably [13]. To reduce the variance in target values, TD3 also adds clipped noise to the selected action to smooth the target policy. There is also a replay memory $\mathcal{B}$ where the agent stores the transitions at each time step after taking an action and samples N transitions in one mini-batch for updating the neural networks during the training.

Handling Varied Number of Local Updates with LSTM

As we have assumed real-world settings of federated learning, the number of local updates received by the server may not be consistent with the number of selected clients at the beginning of the communication round. However, if we were to choose neural networks with a fixed number of neurons as the neural network model in our DRL agent, it may not be able to handle this situation where the N training samples in a mini-batch have different sizes [14]. Inspired by the ability of recurrent neural networks (RNNs) to solve natural language processing tasks and the flexibility to work on sequences of arbitrary lengths, we propose to utilize RNNs as our agent's "brain" to cope with a varied number of client updates. The varied-length sequences at one mini-batch can be fed into RNNs after they are padded to the largest sequence in the batch. We choose one of the state-of-the-art RNN architectures, Long Short-Term Memory Networks (LSTM), as the heads of actor and critic neural networks. Between the LSTM and the final output, there are two full connected layers. The observed state $\mathbf{s}^t$ at each time step t can be regarded as a sequence in natural language processing.

Equipped with LSTM in its neural network architecture, the agent is more flexible when accommodating an arbitrary number of client updates, not only in different time steps within one federated learning session, but also in different federated learning sessions. In other words, the agent's action will not be affected by the potential lack of reliability of the selected clients, and there is no need to re-train the learned DRL model for settings of different numbers of selected clients, making the model more generalizable. In addition, with the use of long-term memory in the LSTM model, the agent is able to interact with FL environments with a history of observations, beyond the current state at any time step.

Overall, Fig. 1 illustrates the workflow of model quality estimation for aggregation. With different number of local updates $\Delta\mathbf{w}_k^t$ along with information w_k^t received by the server in each round t in the asynchronous setting, the DRL agent can generate corresponding evaluation of update quality $\mathbf{a}^t := \mathbf{v}^t$ and aggregate those updates accordingly by $\mathbf{w}^{t+1} = \mathbf{w}^t + \sum_{k\in\mathcal{S}^t} v_k^t \Delta\mathbf{w}_k^t$.

Performance Evaluation

Based on Plato, our open-source federated learning framework, We have implemented Fei, including both the training of the DRL agent, and the deployment of the agent in actual FL sessions. We show the results of our performance evaluations over two benchmark datasets and model combinations: the `FashionMNIST` dataset with the `LeNet-5` model, and the `CIFAR-10` dataset with the `ResNet-18` and `VGG-16` models. We observe the global model accuracy on the test dataset as the number of communication rounds progresses, and the elapsed wall-clock time for the global model to converge.

To evaluate the performance of our approach in comparison with state-of-the-art heuristics that consider the quality of model updates, we compare Fei with the following aggregation mechanisms in federated learning.

FedAvg [1]: As the first aggregation mechanism, FedAvg simply aggregates local model updates according to the proportion of local data size (i.e., number of data samples) to total data size.

AFL [4]: This mechanism estimates the quality of a client model using its local training loss and local data size.

FedAtt [5]: This mechanism measures the quality of a client model using its distance to the global model in terms of layer-wise parameters based on an attention mechanism, which is also adopted by FedDA [10].

FedAdp [6]: This mechanism adaptively measures the quality of a client model using the angle between its gradient and the global gradient.

FedProx [15]: This mechanism is designed to address device heterogeneity across different clients using model regularization in local training.

Training the DRL Agent

In our federated learning sessions used for training the DRL agent, we selected the `FashionMNIST` dataset and the `LeNet-5` model. The client pool has a total of 100 clients, and the server randomly selects 6 to 10 from this pool in each round. The number of data samples across clients is uniformly distributed between 80 and 200. With these settings, we introduce more heterogeneity across the participating clients.

With respect to the neural network model used in our DRL agent, we used LSTM and two consecutive fully connected layers in both actor and critic networks, while each fully connected layer has 256 hidden states. The replay memory in the DRL agent can store 1000 transitions, and each mini-batch has 64 samples. The coefficient in the reward function is set to $\beta = 20$, and the threshold that controls the steady global model stage is set to $\sigma = 0.005$. The DRL agent is trained on a NVIDIA RTX A4500 GPU with 20GB CUDA memory. The time it takes for the training to complete is around 15 hours. In our subsequent experiments, we use this pre-trained DRL model, referred to as Fei, for evaluation and comparison.

Efficiency over Heterogeneous Settings

In the same FL environment where the DRL agent is pre-trained in — the `FashionMNIST` dataset with the `LeNet-5` model — we first investigate the efficiency of our pre-trained DRL model, Fei, under different levels of non-IID data. There are 100 clients in total, and the number of selected clients is set to 10. In each round, the number of training epochs allocated to each selected client is set to 10, and the local training batch size is set to 32. To simulate device heterogeneity, we use the Pareto distribution to simulate the amount of time each client takes to finish each round of local training. To synthesize data heterogeneity, we use the Dirichlet distribution in our experiments. The concentration parameter $\alpha \geq 0$ is used to control the level of data heterogeneity across clients. The data on each client is IID when α ® ∞ and is severely non-IID when α is close to zero. We use a discrete range of α and compare the total wall-clock time for the global model to reach a target test accuracy 82 percent as well as the accuracy at the first round, respectively, as shown in Table 1. In the scenario where α = 0.5*, the number of data samples across clients is uniformly distributed between 80 and 200, which is identical to our FL environment used for training the DRL agent.

Our experimental results show that all alternative mechanisms can improve the accuracy at the beginning over the baseline mechanism, FedAvg, in some scenarios. In terms of the simulated wall-clock time, FedAtt and FedAdp cannot always outperform FedAvg. Overall, AFL, FedProx, and Fei stand out in all scenarios in terms of both performance metrics.

Generalizability and Flexibility

We apply the same DRL model — which we have trained on `FashionMNIST` — onto a different FL session using the `CIFAR-10` dataset with two different CNN models, `ResNet-18` and `VGG16`, to evaluate how Fei is able to generalize to different tasks compared to what it was trained on. For each selected client, the allocated number of training epochs is 5, and the training batch size is set to 64 for `ResNet-18` and 128 for `VGG16`. Figure 2

Algorithms	non-IID	Wall-clock time	1st-round Accuracy
FedAvg	$\alpha = \infty$	8803	54.8
AFL		1402	66.6
FedAtt		10467	57.6
FedAdp		7443	53.9
FedProx		2640	73.8
FEI		653	78.4
FedAvg	$\alpha = 0.5$	4360	37.8
AFL		4187	62.4
FedAtt		3700	38.1
FedAdp		5740	32.6
FedProx		1660	57.4
FEI		2291	64.6
FedAvg	$\alpha = 0.5^*$	11853	22.9
AFL		6886	42.0
FedAtt		11198	33.4
FedAdp		13576	23.8
FedProx		6084	57.0
FEI		2974	53.9

TABLE 1. The elapsed wall-clock time for the global model to achieve the target test accuracy as well as the test accuracy at the first round, with different levels of non-IID data over the `FashionMNIST` dataset with the `LeNet-5` model.

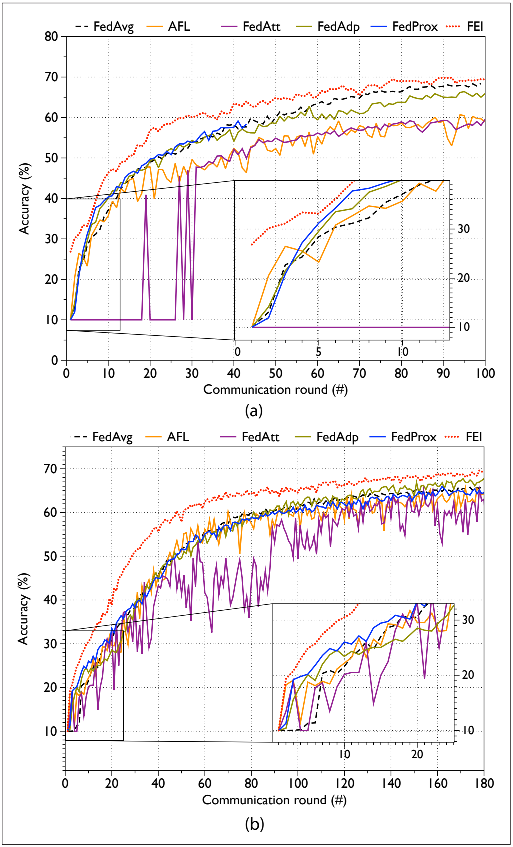

FIGURE 2. Test accuracy over communication rounds of different FL algorithms on `CIFAR-10` dataset: a) with `ResNet-18`; b) with `VGG-16`.

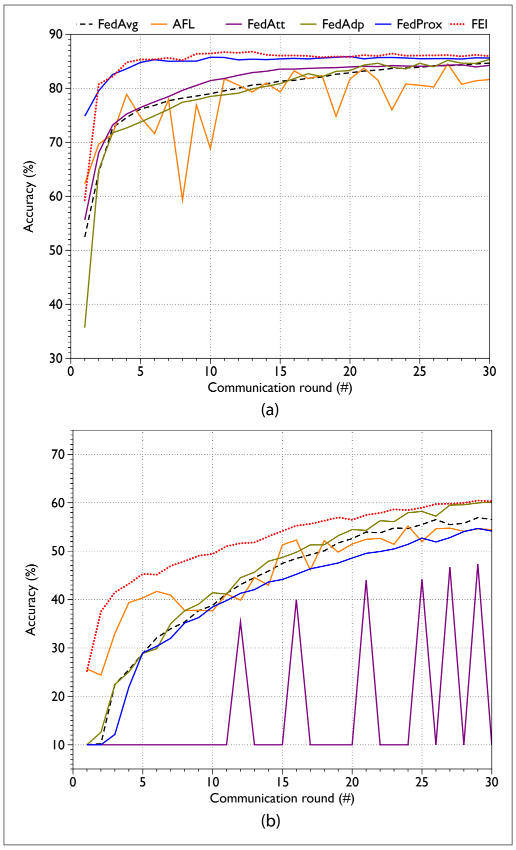

FIGURE 3. Test accuracy over communication rounds of different FL algorithms in a larger cohort of clients: a) on `FashionMNIST` with `LeNet-5`; b) on `CIFAR-10` with `ResNet-18`.

shows the test accuracy of the global model vs. the number of communication rounds in the FL session.

We can observe that FEI outperforms all other federated learning mechanisms in terms of the convergence speed, even though the DRL model was not trained in the same environments. This shows strong evidence that, with appropriate pre-training, FEI can be utilized in different federated learning tasks to shorten the time of training a global model with high quality. We argue that the observed generalizability of our DRL agent can be attributed to our design on how observable states can be represented, so that the quality of client updates can be properly evaluated. In contrast, FedAtt suffered from a worrisome lack of stability in its convergence behavior, which may be due to the complexity of its neural network architecture, which may introduce more bias or variance in the aggregation weights due to the layer-wise attentive mechanism. This shows that a suitable design of the neural network architecture is of utmost importance when neural network models are trained and tasked to make on-the-fly decisions in an FL session.

Last but not the least, we investigate the flexibility of FEI in a scenario where it faces a larger scale of clients. Rather than selecting 10 clients from 100 as in DRL agent training, we select 50 each round from 200 clients. Other settings are the same as our previous experiments. As shown in Fig. 3a, FEI and FedProx can still converge very quickly on `FashionMNIST` within 10 rounds, while FEI achieves a slightly higher ultimate accuracy. The convergence trend in Fig. 3b indicate that on `CIFAR-10`, the global model with FEI will potentially have an ultimate accuracy lower than FedAvg's and FedAdp's. But the bright side is that the accuracy of FEI at the beginning of the session is still higher than its alternatives, and it keeps growing steadily, unlike AFL and FedAtt. In contrast, for example, the accuracy with AFL fluctuates significantly in most experimental scenarios.

Conclusion

In this article, we investigate the quantification of local update quality for federated learning in heterogeneous settings closer to real-world communication networks. We propose FEI, a federated learning framework with a novel aggregation strategy to actively evaluate the local contributions, aiming to enhance the ultimate global model quality and to minimize the number of communication rounds or even the wall-clock time needed for convergence. In particular, FEI leverages a DRL-based agent that applies the TD3 algorithm with LSTM neural networks to determine the quality of each local update. By comparing our method with FedAvg as well as other state-of-the-art FL algorithms with different quality evaluation strategies on multiple FL training tasks, we show that FEI has efficiently reduced the amount of wall-clock time needed to reach the same target accuracy on FashionMNIST and `FashionMNIST` datasets with different levels of non-IID data, and has exhibited the great potential of adapting to FL tasks different from what the DRL agent has been trained on. In addition, the LSTM-TD3 architecture in FEI allows it to adaptively aggregate an arbitrary number of client updates according to their qualities. A limitation of this study is that a DRL model pre-trained on one FL environment may not optimally generalize many other FL environments. A natural progression of this work is to apply continual learning to the proposed design to learn a model for a broader range of FL tasks without forgetting previous knowledge.

Acknowledgments

The research was supported in part by RGC RIF grant R6021-20, and RGC GRF grants under the contracts 16209120 and 16200221.

References

[1] B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks From Decentralized Data," *Proc. Int'l. Conf. Artificial Intelligence and Statistics*, 2017, pp. 1273–82.
[2] A. Hard *et al.*, "Federated Learning for Mobile Keyboard Prediction," arXiv preprint arXiv:1811.03604, 2018.
[3] S. R. Pfohl, A. M. Dai, and K. Heller, "Federated and Differentially Private Learning for Electronic Health Records," arXiv preprint arXiv:1911.05861, 2019.
[4] J. Goetz *et al.*, "Active Federated Learning," arXiv preprint arXiv:1909.12641, 2019.
[5] S. Ji *et al.*, "Learning Private Neural Language Modeling With Attentive Aggregation," *Proc. Int'l. Joint Conf. Neural Networks (IJCNN)*, 2019, pp. 1–8.
[6] H. Wu and P. Wang, "Fast-Convergent Federated Learning With Adaptive Weighting," *IEEE Trans. Cognitive Commun. and Networking*, 2021, pp. 1–1.
[7] M. Tang *et al.*, "FedGP: Correlation-Based Active Client Selection for Heterogeneous Federated Learning," arXiv preprint arXiv:2103.13822, 2021.
[8] S. R. Pandey, L. D. Nguyen, and P. Popovski, "A Contribution-Based Device Selection Scheme in Federated Learning,"

IEEE Commun. Letters, 2022.

[9] M. H. ur Rehman *et al.*, "Towards Blockchain-Based Reputation-Aware Federated Learning," *IEEE Conf. Computer Commun. Workshops (INFOCOM WKSHPS)*, 2020, pp. 183–88.

[10] C. Zhang *et al.*, "Dual Attention-Based Federated Learning for Wireless Traffic Prediction," *Proc. IEEE INFOCOM*, 2021.

[11] H. Wang *et al.*, "Optimizing Federated Learning on Non-IID Data With Reinforcement Learning," *Proc. IEEE INFOCOM*, 2020, pp. 1698–1707.

[12] Y. Zhan and J. Zhang, "An Incentive Mechanism Design for Efficient Edge Learning by Deep Reinforcement Learning Approach," *Proc. IEEE INFOCOM*, 2020, pp. 2489–98.

[13] S. Fujimoto, H. Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *Proc. Int'l. Conf. Machine Learning. PMLR*, 2018, pp. 1587–96.

[14] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[15] T. Li *et al.*, "Federated Optimization in Heterogeneous Networks," *Proc. Machine Learning and Systems*, vol. 2, 2020, pp. 429–50.

Biographies

Fei Wang is currently pursing her Ph.D. at the Department of Electrical and Computer Engineering, University of Toronto, Canada. She received her B.Eng. degree in Computer Science and Technology at Hongyi Honor College, Wuhan University. Her research interests lie at the intersections of reinforcement learning and computer communications.

Baochun Li received his B.Eng. degree from Tsinghua University and his M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign. He is a professor in the Department of Electrical and Computer Engineering, University of Toronto. His research interests include large-scale distributed systems, cloud computing, and wireless networks.

Bo Li received his B.Eng. and M.Eng. degrees in Computer Science from Tsinghua University, and a Ph.D. degree in Electrical and Computer Engineering from the University of Massachusetts at Amherst. He is a chair professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology.