Resource and Heterogeneity-Aware Personalized Federated Learning via Single-Agent Reinforcement Learning

Kavindu Herath, Don Alwis, Nethmi Hewa Withthige, Lakshika Karunaratne

December 18, 2025

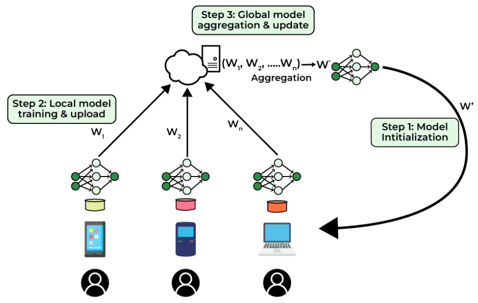
1 Introduction

1.1 Literature Review

Federated Learning (FL) is a distributed machine learning paradigm that enables multiple clients to collaboratively train a shared model without directly exchanging their raw data [1, 2]. Instead of transferring sensitive local datasets to a central server, each client performs model training locally using its private data and only shares model updates such as gradients or model parameters with a central coordinator. The server aggregates these updates to construct a global model, which is then redistributed to clients for subsequent training rounds. This decentralized training framework significantly enhances data privacy, reduces the communication of raw data, and makes FL well suited for privacy-sensitive and large-scale distributed environments such as healthcare systems, financial institutions, smart cities, Internet-of-Things (IoT) networks, and recommendation systems in mobile applications.

FL faces several practical challenges, among which data heterogeneity is one of the most critical. In real-world scenarios, clients often generate data from diverse sources, user behaviors, or operating conditions, leading to data distributions that vary significantly across clients. This heterogeneity can degrade the convergence speed and performance of a single global model, as it may fail to generalize well across all participants. To address this limitation, Personalized Federated Learning (pFL) has emerged as an extension of FL that aims to provide customized models tailored to individual clients [3, 4]. Rather than enforcing a single global model for all users, pFL allows each client to obtain a personalized model that better reflects its local data characteristics, while still benefiting from shared knowledge across the federation. Broadly, existing pFL methods can be categorized into two main types. The first type achieves personalization through partial parameter sharing, where a common global model is learned across clients, while client-specific parameters are maintained locally to capture individual data characteristics [5]. The second type focuses on learning separate personalized models for each client through coordinated optimization, rather than sharing a fixed global model [6, 7]. In this setting, clients collaborate to learn a shared initialization, regularization, or optimization strategy, which is then adapted locally to obtain fully personalized models.

Reinforcement Learning (RL) has recently been introduced as a powerful tool to enhance both FL and pFL systems by enabling adaptive and intelligent decision-making [8–10]. FL

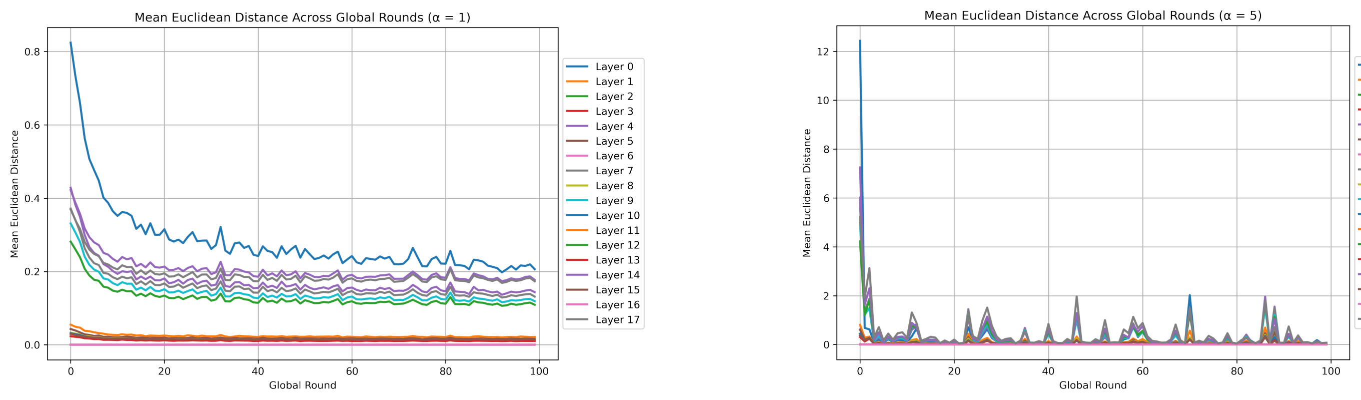

Step 3: Global model aggregation & update
$(W_1, W_2, \dots W_n) \rightarrow W'$
Aggregation
Step 2: Local model training & upload
W_1
W_2
W_n
W'
Step 1: Model Intitialization

(a) Overview of the federated learning (FL) workflow

environments involve complex trade-offs among multiple factors, including communication overhead, limited computational resources, and data diversity. RL provides a principled framework to dynamically optimize these decisions based on system feedback and long-term performance objectives. In existing studies, RL has been successfully applied to tasks such as client selection [9], adaptive allocation of local training intensity [11], and dynamic model size or structure allocation[12]. These RL-driven approaches help improve training efficiency and personalization performance in personalized federated learning systems.

1.2 Motivation

Most existing approaches that apply RL to FL and pFL rely on multiple agents or decentralized control mechanisms [13–15]. In these methods, each client (or a group of clients) is treated as an independent RL agent that makes local decisions, such as whether to participate in a training round, how many local epochs to run, or how much of the model to transmit. These agents typically operate based on their own observations and may coordinate through the shared global model or through message passing. While multi-agent RL provides flexibility and scalability, it introduces additional challenges, such as non-stationary learning dynamics, increased coordination complexity, and higher communication overhead. In contrast, there is only limited work on using a single centralized RL agent to coordinate the entire federated learning process while jointly considering both data heterogeneity and client computational capabilities. Motivated by this gap, our project aims to design a single-agent reinforcement learning framework that dynamically determines which portions of the model are trained locally and which layers are transmitted to the server by each client, based on their computational ca-



(a) Mean Euclidean Distance, $\alpha = 1$ (b) Mean Euclidean Distance, $\alpha = 5$

Figure 2: Layer-wise mean Euclidean distances across clients over global rounds for different Dirichlet α values.

pacity, bandwidth constraints, and the test accuracies of both global and local models.

1.3 Objective

To design a reinforcement learning approach that improves the accuracy and efficiency of personalized federated learning across heterogeneous clients with varying data distributions and computational capabilities.

2 Background

2.1 Federated Learning

Federated learning (FL) is a distributed learning paradigm in which multiple clients collaboratively train a model under the coordination of a central server, while keeping all raw data on-device. The most widely used baseline in FL is *Federated Averaging (FedAvg)* [16], which learns a single global model by repeatedly alternating between local client optimization and server-side aggregation.

Assume there are K clients. Client k holds a local dataset P_k with n_k samples, and $n = \sum_{k=1}^{K} n_k$ is the total number of samples across all clients. The global learning objective can be written as

$$\min_{w} f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{1}$$

$$F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} \ell(w; x_i, y_i) \tag{2}$$

where w denotes the model parameters, $F_k(w)$ is the local objective at client k, and $\ell(w; x_i, y_i)$ is the per-sample loss (e.g., cross-entropy for classification).

At communication round t, the server broadcasts the current global model w_t to a subset of clients S_t. Each selected client performs local training (typically multiple SGD steps over its private data) starting from w_t to obtain an updated local model w_{t+1}^k. The server then aggregates the local models using a weighted average:

$$w_{t+1} = \sum_{k \in S_t} \frac{n_k}{\sum_{j \in S_t} n_j} w_{t+1}^k \tag{3}$$

By allowing clients to execute multiple local updates before aggregation, FedAvg reduces communication frequency compared to federated SGD, and it serves as a standard baseline for evaluating FL systems under heterogeneous and non-IID data distributions.

2.2 Personalized Federated Learning

Personalized Federated Learning (pFL) extends standard federated learning by allowing each client to learn a model tailored to its own data, while still leveraging information from a global shared model. This is particularly useful when client data distributions are heterogeneous (non-IID). Assume there is a set of clients $\mathcal{U}$. Each client $k \in \mathcal{U}$ has a local dataset D_k. The goal of pFL is to learn personalized model parameters θ_k for each client while maintaining a global model θ_g. A general objective is:

$$\min_{\theta_k,\, k \in \mathcal{U}} \sum_{k \in \mathcal{U}} \mathbb{E}_{(x,y) \sim D_k} \Big[\mathcal{L}(F(x; \theta_k), y) + \mathcal{R}(\theta_k, \theta_g) \Big] \tag{4}$$

where: $\mathcal{L}(F(x; \theta_k), y)$ is the local loss measuring how well the personalized model θ_k predicts the label y for input x. $\mathcal{R}(\theta_k, \theta_g)$ is a regularization term that encourages θ_k to stay close to the global model θ_g, improving generalization on unseen data.

At each communication round t, clients update their personalized models using local SGD:

$$\theta_k^{t+1} = \theta_k^t - \eta \nabla_{\theta_k} \Big[\mathcal{L}(F(x; \theta_k^t), y) + \mathcal{R}(\theta_k^t, \theta_g^t) \Big] \tag{5}$$

The server can optionally update the global model θ_g via aggregation:

$$\theta_g^{t+1} = \sum_{i \in S_t} \frac{|D_i|}{\sum_{j \in S_t} |D_j|} \theta_k^{t+1} \tag{6}$$

Here, $S_t \subseteq \mathcal{U}$ is the set of clients selected at round t, and $|D_k|$ denotes the number of samples in client k's dataset.

This framework allows each client to maintain a personalized model while still benefiting from knowledge shared across clients, improving performance on heterogeneous datasets. More details can be found in [7][6].

2.3 RL-based Personalized Federated Learning

Reinforcement learning (RL) has emerged as an effective tool for improving decision-making in federated learning (FL) and personalized federated learning (pFL), particularly in settings characterized by heterogeneous data distributions, variable client resources, and communication constraints. In RL-based FL, the training process is commonly formulated as a sequential or contextual decision-making problem, where agents observe system states such as client accuracies, model divergence, resource availability, or participation history and take actions that influence the training dynamics, including client selection, local training intensity, or model update scheduling [8, 9, 11].

Existing RL-based pFL approaches predominantly rely on *multi-agent* or *decentralized* learning paradigms. In these methods, each client, or a subset of clients, is modeled as an independent RL agent that makes local decisions based on its own observations and objectives [13–15].

Despite the growing body of work on RL-driven pFL, there is limited prior research that explores the use of a *single centralized RL agent* to coordinate federated learning decisions across all clients. The potential benefits of centralized RL control such as global visibility of client behavior, reduced coordination overhead, and simpler learning dynamics remain largely underexplored in the context of personalized federated learning.

3 Methodology

3.1 Defining action space for layer selection

We analyze the layer-wise behavior of client models during federated learning on CIFAR-10 with 10 clients, using Dirichlet splits ($\alpha = 1$ and $\alpha = 5$) to simulate heterogeneous data. Each client performs 1 epoch of local training per round over 100 global rounds.

After each local update, the Euclidean distance between each client's layers and the corresponding layers of the global model is computed, and these distances are then averaged across all clients for each layer at every communication round. The resulting averaged plots (Figure 2) reveal how layers diverge or converge on average, providing insight into layer-wise training dynamics and the impact of data heterogeneity. A comparison across different α values further illustrates how increasing heterogeneity influences client model evolution and the overall behavior of federated learning across layers.

As shown in Figure 2, the mean Euclidean distance across clients is concentrated in the first k layers, while the remaining layers exhibit minimal variation. Accordingly, the first k layers are adopted as base layers and the remaining layers as personalization layers, following the principle of FedPer [17]. This design allows early layers to capture generalizable features through global collaboration, while later layers adapt to client-specific data, enabling effective personalization under statistical heterogeneity and reducing communication overhead.

3.2 Defining action space for training intensity

Training intensity, referring to the number of local training epochs executed by a client in each global communication round, is a key factor in our work and directly affects both performance and computational cost. While training intensity is commonly treated as an integer-valued parameter, restricting it to integers may limit flexibility when defining the action space of a learning agent. To study whether integer-valued local epochs are sufficient or whether fractional epochs are required, we conducted experiments with 10 clients under two non-IID data distributions. For each client, the local dataset was split into training and test subsets using a 19:1 ratio.

Figures 3c and 3 present representative results for selected clients. In the less non-IID setting 3c, 0.5 and 1 local epochs achieve nearly identical global and local test accuracies, indicating that fractional epochs can provide similar performance at lower computational cost. In contrast, under a highly non-IID distribution, higher integer-valued local epochs lead to improved local and global test accuracies. These results suggest that an action space containing both fractional and integer local epochs enables more efficient and adaptive control of training intensity under varying data heterogeneity and computational constraints.

3.3 Reinforcement Learning Approach

We adopt a single server-side reinforcement learning (RL) agent to jointly control client participation behavior in personalized federated learning. The agent operates centrally at the server and makes client-specific decisions at every communication round, enabling adaptive control under data heterogeneity and system-level resource constraints.

RL Formulation. The federated learning process is formulated as a *contextual bandit* problem, where each communication round is treated as an independent decision step. For each client k at round t, the RL agent observes a four-dimensional state vector

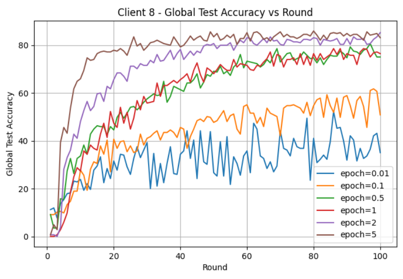
$$s_k^t = \left[\mathrm{emb}(i),\ a_k^{t-1},\ b_k,\ e_k\right] \tag{7}$$

where $\mathrm{emb}(i)$ identifies the client using one-hot encoding, a_k^{t-1} denotes the previous-round local training accuracy, b_k represents the client's communication budget (e.g., maximum allowable number of shared layers), and e_k captures the client's preferred or allowable local training intensity (number of epochs). This state jointly encodes both statistical heterogeneity and resource constraints.

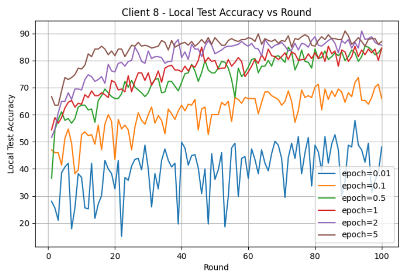
Action Space. Based on the observed state, the agent selects a joint action

$$a_k^t = (L_k^t, E_k^t) \tag{8}$$

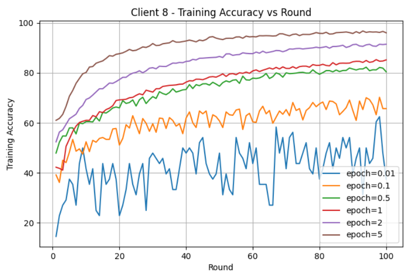
where $L_k^t \in \{1, \ldots, L\}$ denotes the number of model layers client k shares with the server, and $E_k^t \in \mathcal{E}$ denotes the local training intensity (fractional number of epochs) assigned to the client for the next round. This joint action directly determines both communication cost and computational effort.



(a) Global test accuracy

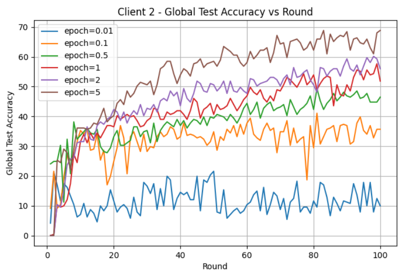


(b) Local test accuracy

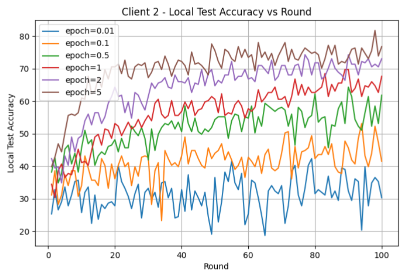


(c) Training accuracy

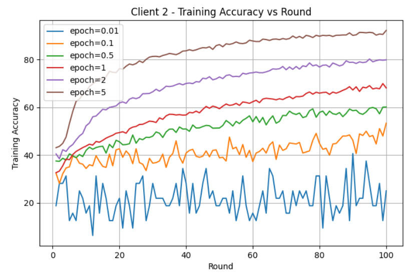
Figure 3: Accuracies of Client 8 across communication rounds under a less non-IID data distribution. The client contains 5713 samples (11.43% of the total data). Shown are (a) global model test accuracy evaluated on the client's test data, (b) local model test accuracy before aggregation, and (c) local model training accuracy.



(a) Global test accuracy



(b) Local test accuracy



(c) Training accuracy

Figure 4: Accuracies of Client 2 across communication rounds under a less non-IID data distribution. Shown are (a) global model test accuracy evaluated on the client's test data, (b) local model test accuracy before aggregation, and (c) local model training accuracy.

Learning Algorithm. We employ an ϵ-greedy neural Q-learning algorithm tailored to the contextual bandit setting. A neural network parameterized by θ approximates the action-value function $Q_\theta(s, a)$, which estimates the immediate reward associated with taking action a in state s. At each decision step, the agent selects

$$a_k^t = \begin{cases} \text{random action}, & \text{with probability } \epsilon \\ \arg\max_{a \in \mathcal{A}} Q_\theta(s_k^t, a), & \text{otherwise} \end{cases} \tag{9}$$

Reward and Optimization. After executing the selected action and completing the corresponding federated update, the agent receives an immediate reward r_k^t (Section 3.6) that reflects model performance and constraint compliance. Since rewards are observed within the same round, no temporal bootstrapping or discounting is required. The Q-network is trained via regression to the observed reward using the Huber (Smooth-ℓ_1) loss:

$$\mathcal{L}(\theta) = \text{Huber}\big(Q_\theta(s_k^t, a_k^t) - r_k^t\big) \tag{10}$$

This formulation enables stable online learning while remaining robust to noisy reward signals.

Based on the empirical analysis presented in the preceding experimental studies (Section 4.3), we define the action space of the reinforcement learning agent in a principled manner. Specifically, the layer-wise distance analysis and training-intensity experiments provide quantitative evidence on how layer sharing and local training effort influence model divergence, generalization, and computational cost. Guided by these observations, we design an action space that captures the most influential control dimensions while remaining interpretable and efficient. For a methodical evaluation, we decompose the overall approach into three cases: (i) an RL agent that controls only the number of shared layers (Section 3.4), (ii) an RL agent that controls only the local training intensity (Section 3.5), and (iii) a combined RL agent that jointly controls both layer sharing and training intensity (Section 3.6). Each case is analyzed separately in the following sections to isolate their individual effects before evaluating their combined behavior.

3.4 Layer Selection Control via an RL Agent

We consider the same cross-device federated learning (FL) setup with K clients and a central server. Training proceeds in communication rounds $t = 1, 2, \ldots$. In each round, the server broadcasts the current global model to the selected clients, clients perform local training, and the server aggregates returned updates to obtain the next global model.

To make communication resource-aware while enabling personalization, we learn a server-side policy that assigns each client a *sharing depth* per round, represented as the number of model layers to share with the server. Each client

Table 1: Per-client preferred local training capacity and communication budget.

Client ID	0	1	2	3	4	5	6	7	8	9
Computational Capacity	0.5	0.01	1.0	2.0	2.0	0.5	1.0	2.0	0.1	5.0
Communication Budget	3	5	1	2	4	6	3	6	2	5

k is associated with a preferred communication budget b_k (preferred number of layers to exchange per round), which we treat as a soft constraint and penalize when exceeded.

We model layer selection as a contextual bandit executed on the server. For each participating client k in round t, the agent observes a state vector

$$s_k^t = \big[\mathrm{emb}(k)\,;\ a_k^{t-1}\,;\ b_k\big], \tag{11}$$

where $\mathrm{emb}(k)$ is a client-ID encoding, a_k^{t-1} is the client's recent local training accuracy, and b_k is the preferred layer-sharing budget. The agent selects an action $\ell_k^t \in \mathcal{A}_\ell$, where $\mathcal{A}_\ell = \{1, 2, \ldots, L\}$ and L is the total number of sharable layers. The selected action ℓ_k^t specifies that client k shares the *first* ℓ_k^t layers (layers 1 through ℓ_k^t), while the remaining layers are kept local to preserve personalization and reduce communication.

After receiving client updates, the server performs *partial aggregation* over the shared layers only. Specifically, for each layer index $\ell \leq \ell_k^t$, the server aggregates parameters from clients that shared that layer in round t using a FedAvg-style weighted average. Layers that are not shared by a client are not uploaded and are not aggregated.

The server computes an immediate reward that balances performance and communication compliance:

$$r^t = \sum_{k \in \mathcal{S}^t} a_k^t \;-\; \lambda_{\text{comm}} \sum_{k \in \mathcal{S}^t} \max(0,\, \ell_k^t - b_k) \;-\; \lambda_{\text{dist}} \sum_{k \in \mathcal{S}^t} d_k^t, \tag{12}$$

where $\mathcal{S}^t$ is the set of participating clients, a_k^t is the client's local training accuracy, the second term penalizes exceeding the preferred layer budget, and d_k^t measures divergence of client k's shared update from others (e.g., an ℓ_2 distance to the mean shared update over layers 1 through ℓ_k^t).

We use an ϵ-greedy contextual bandit with a neural Q-function $Q_\theta(s, a)$. With probability ϵ the agent explores a random action; otherwise it selects $a = \arg\max_{a \in \mathcal{A}_\ell} Q_\theta(s_k^t, a)$. Since feedback is round-local, we directly regress the Q-network onto the observed immediate reward using the Huber (Smooth-ℓ_1) loss:

$$\mathcal{L}(\theta) = \mathrm{Huber}\big(Q_\theta(s_k^t, \ell_k^t) - r^t\big). \tag{13}$$

We update θ via stochastic optimization.

3.5 Training Intensity Control via an RL Agent

We consider a standard cross-device federated learning (FL) setup with N clients and a central server. Training proceeds in communication rounds $t = 1, 2, \ldots$. In each round, the server broadcasts the current global model to the selected clients, clients perform local training, and the server aggregates returned updates in a FedAvg-style manner to obtain the next global model.

To make training resource-aware, we learn a server-side policy that assigns each client a *training intensity* per round, represented as a fractional number of local epochs. Each client i is associated with a preferred local training budget c_i (preferred epochs per round), which we treat as a soft constraint and penalize when exceeded.

We model intensity selection as a contextual bandit executed on the server. For each participating client i in round t, the agent observes a state vector

$$s_i^t = \big[\mathrm{emb}(i)\,;\ a_i^{t-1}\,;\ c_i\big], \tag{14}$$

where $\mathrm{emb}(i)$ is a client-ID encoding, a_i^{t-1} is the client's recent local training accuracy, and c_i is the preferred epoch budget. The agent selects an action $e_i^t \in \mathcal{A}_e$, where $\mathcal{A}_e$ is a discrete set of epoch fractions (e.g., $\{0.1, 1.0, 2.0, \ldots\}$). Fractional epochs are implemented by training for a proportional number of mini-batches from the local data loader.

After local training, the server computes an immediate reward that balances learning progress and constraint compliance:

$$r^t = \sum_{i \in \mathcal{S}^t} a_i^t \;-\; \lambda_{\text{comp}} \sum_{i \in \mathcal{S}^t} \max(0,\, e_i^t - c_i) \;-\; \lambda_{\text{dist}} \sum_{i \in \mathcal{S}^t} d_i^t, \tag{15}$$

where $\mathcal{S}^t$ is the set of participating clients, a_i^t is the client's local training accuracy, and d_i^t measures how far client i's update is from others (e.g., an ℓ_2 distance to the mean update over shared parameters).

We use an ϵ-greedy contextual bandit with a neural Q-function $Q_\theta(s, a)$. With probability ϵ the agent explores a random action; otherwise it selects $a = \arg\max_{a \in \mathcal{A}_e} Q_\theta(s_i^t, a)$. Since feedback is round-local, we directly regress the Q-network onto the observed immediate reward using the Huber (Smooth-ℓ_1) loss:

$$\mathcal{L}(\theta) = \mathrm{Huber}\big(Q_\theta(s_i^t, e_i^t) - r^t\big) \tag{16}$$

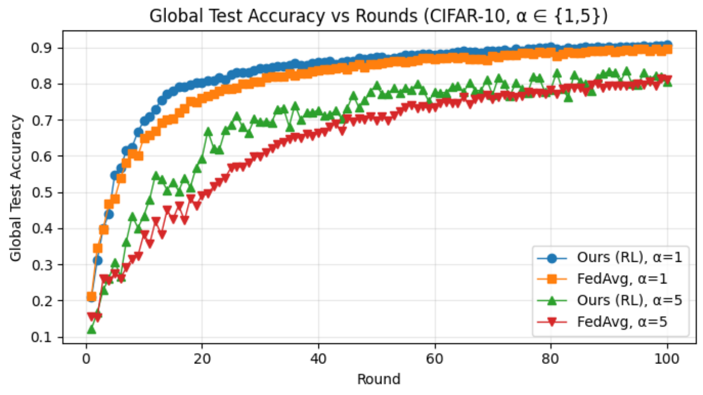
3.6 Single Agent Controlling Both Layer Selection and Training Intensity

In addition to studying (i) training-intensity control and (ii) shared-layer selection as two separate components, we also design a *single* server-side RL agent that jointly makes both decisions for each client in every communication round. The goal is to simultaneously account for heterogeneous computation and communication preferences while maintaining strong model performance.

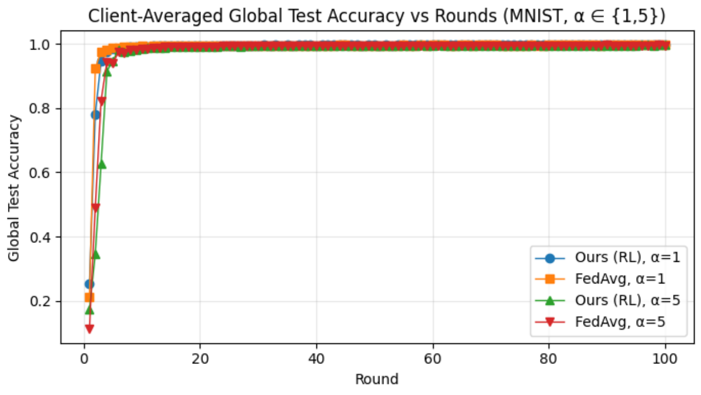
We assume each client i has two preference parameters: a preferred local training budget c_i (preferred epochs per

Table 2: Comparison of baseline (FedAvg-style) and our method for computational capacity (Comp. Cap) and communication budget (Comm. Bud.) exceedance under different datasets and Dirichlet heterogeneity levels.

Method	Violations	CIFAR-10				MNIST			
		$\alpha = 1$		$\alpha = 5$		$\alpha = 1$		$\alpha = 5$	
		Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours
Layer Selection	Comm. Bud. Exceeding	400	**351**	400	**359**	400	**361**	400	**346**
Training Intensity	Comp. Cap. Exceeding	400	**332**	400	**339**	400	**352**	400	**336**
Layer Selection	Comp. Cap. Exceeding	400	**296**	400	**322**	400	**331**	400	**300**
+ Training Intensity	Comm. Bud. Exceeding	400	**321**	400	**396**	400	**353**	400	**386**



(a) CIFAR-10: Global test accuracy over communication rounds.



(b) MNIST: Global test accuracy over communication rounds.

Figure 5: Comparison of our RL-based client scheduling policy and standard FedAvg under two Dirichlet non-IID settings ($\alpha = 1$ and $\alpha = 5$).

round) and a preferred communication budget ℓ_i (preferred number of layers to share), where $\ell_i \in \{1, \ldots, L\}$ and L denotes the total number of sharable layers. At round t, for each participating client i, the agent observes the client-specific state

$$s_i^t = \big[\mathrm{emb}(i)\,;\ a_i^{t-1}\,;\ c_i\,;\ \ell_i\big], \tag{17}$$

where $\mathrm{emb}(i)$ encodes the client identity and a_i^{t-1} is the client's recent local training accuracy.

The agent outputs a *joint action* consisting of two decisions:

$$e_i^t \in \mathcal{A}_e, \qquad \ell_i^t \in \mathcal{A}_\ell, \tag{18}$$

where $\mathcal{A}_e$ is the discrete set of fractional epoch choices and $\mathcal{A}_\ell = \{1, 2, \ldots, L\}$ is the layer-sharing action space. Here e_i^t specifies the assigned local training intensity and ℓ_i^t specifies the sharing depth, meaning that client i shares the **first** ℓ_i^t layers (layers 1 through ℓ_i^t) with the server in that round.

Given (e_i^t, ℓ_i^t), client i performs local training for e_i^t epochs (implemented via a proportional number of mini-batches) and uploads only the parameters corresponding to layers 1 through ℓ_i^t. The server aggregates only these shared layers across participating clients (FedAvg-style), while the remaining layers are kept local and are not aggregated, enabling personalization and reducing communication.

After observing the round outcomes, the server computes an immediate reward that encourages accuracy while penalizing violations of both compute and communication preferences and discouraging outlier updates:

$$\begin{aligned} r^t = \sum_{i \in \mathcal{S}^t} a_i^t - \lambda_{\text{comp}} \sum_{i \in \mathcal{S}^t} \max(0, e_i^t - c_i) \\ - \lambda_{\text{comm}} \sum_{i \in \mathcal{S}^t} \max(0, \ell_i^t - \ell_i) - \lambda_{\text{dist}} \sum_{i \in \mathcal{S}^t} d_i^t. \end{aligned} \tag{19}$$

where $\mathcal{S}^t$ is the set of participating clients, a_i^t is the client's local training accuracy, and d_i^t measures how far client i's shared update is from others (e.g., an ℓ_2 distance to the mean shared update).

We train the combined controller using the same contextual bandit approach as in the individual components, with ϵ-greedy exploration and a neural Q-function $Q_\theta(s, a)$. For each client-round interaction, the agent observes s_i^t, selects the joint action $a_i^t = (e_i^t, \ell_i^t)$, and receives the immediate reward r^t. We update the Q-network by regressing the predicted value $Q_\theta(s_i^t, a_i^t)$ toward the observed reward using the Huber (Smooth-ℓ_1) loss:

$$\mathcal{L}(\theta) = \mathrm{Huber}\big(Q_\theta(s_i^t, (e_i^t, \ell_i^t)) - r^t\big)\,. \tag{20}$$

4 Results and Discussion

4.1 Datasets

We evaluate on MNIST and CIFAR-10. MNIST contains 70,000 grayscale digit images (28×28), with 60,000 training and 10,000 test samples across 10 classes. CIFAR-10 contains 60,000 RGB images (32×32) across 10 classes, with

50,000 training and 10,000 test samples. We use the standard train/test splits, apply normalization for both datasets, and use lightweight augmentation for CIFAR-10.

4.2 Experimental Setup

4.2.1 Compute environment

Experiments were conducted using PyTorch, with some runs on Google Colab Pro using a single NVIDIA T4 GPU and others on an NVIDIA A100 80GB GPU.

4.2.2 Non-IID data partitioning

We simulate statistical heterogeneity using label-skew partitions drawn from a Dirichlet distribution with concentration α. We report results for $\alpha \in \{1, 5\}$, where $\alpha = 1$ induces stronger client heterogeneity than $\alpha = 5$. Global evaluation is performed on the standard test set; client-level signals for the RL agent are computed on each client's held-out split (Section 3).

4.2.3 Clients, resource preferences, and policies

We consider 10 clients and a central server. Each client i has a computational preference c_i (fractional epochs per round) and a communication preference b_i (preferred number of shared layers). Table 1 lists these budgets. We treat these budgets as *soft constraints* rather than hard caps (e.g., clipping), and penalize exceedance in the RL reward. This design allows the controller to occasionally violate a client's preferred limits when doing so is beneficial for learning. For example, allocating more local training to clients whose data is more informative, or increasing the shared depth in rounds where sharing more of the model yields a clear accuracy gain. In contrast, hard clipping would prevent such trade-offs entirely and can lead to suboptimal accuracy when strict adherence to preferences conflicts with effective training.

In contrast, our approach uses a single server-side contextual-bandit agent to select, per client and per round, (i) training intensity, (ii) shared-layer depth, or (iii) both jointly, allowing the system to reduce repeated violations while maintaining model performance.

4.2.4 Evaluation metrics

We report (i) client-averaged global test accuracy and (ii) budget exceedance counts for compute ($e_i^t > c_i$) and communication ($\ell_i^t > b_i$).

Hyperparameters. Unless otherwise noted, all experiments use 10 clients trained for 100 global communication rounds with batch size 64. Clients optimize locally with learning rate 0.01, and the server-side RL agent is trained with learning rate 0.001 using an ϵ-greedy policy with initial exploration rate $\epsilon = 1.0$ (decayed over rounds). For non-IID partitioning, we use Dirichlet label-skew splits with $\alpha = 1$ and also report results for $\alpha = 5$. The training-intensity controller selects from fractional-epoch actions $\{0.01, 0.1, 0.5, 1, 2, 5\}$, and the layer-selection controller selects a sharing depth from $\{1, 2, 3, 4, 5, 6\}$. Compute and communication budget violations are penalized with $\lambda_{\text{comp}} = 1.0$ and, for CIFAR-10, $\lambda_{\text{comm}} = 2.0$. The same hyperparameter settings are used across the training-intensity, layer-selection, and combined-controller experiments unless stated otherwise.

4.3 Results

4.3.1 Training intensity control

Figure 5 compares RL-based training intensity selection against FedAvg on MNIST and CIFAR-10 under $\alpha \in \{1, 5\}$. Overall, the RL policy achieves similar (often slightly higher) client-averaged test accuracy while reducing compute-budget violations.

On MNIST, both methods converge quickly, so accuracy differences are small; nevertheless, the RL agent adapts per-client intensity to avoid repeatedly exceeding limited compute budgets. On CIFAR-10, the benefit of adaptive intensity is clearer, particularly under stronger heterogeneity ($\alpha = 1$). In this setting, fixed training intensity can over-allocate computation to constrained clients and exacerbate client drift. By contrast, the RL policy allocates larger training budgets selectively and reduces training when the expected accuracy gain is small. Consistent with this, Table 2 shows that compute-budget exceedances drop from 400 (FedAvg) to 332–352 with RL, depending on dataset and α.

4.3.2 Layer-sharing (communication) control

Figure 6 evaluates RL-based layer selection against FedAvg. The agent chooses how many layers to share per round to balance accuracy and communication compliance under non-IID data.

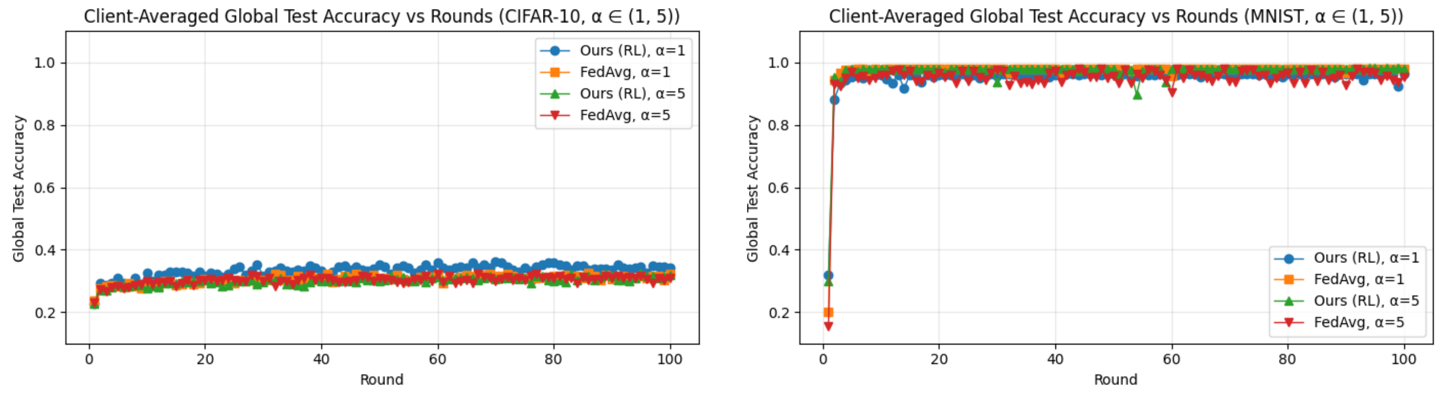
For MNIST, both methods reach high accuracy rapidly, with RL often producing smoother learning under $\alpha = 1$. For CIFAR-10, improvements are more pronounced and are largest under $\alpha = 1$, consistent with partial sharing reducing negative transfer when clients are highly heterogeneous. In terms of compliance, Table 2 shows that communication-budget exceedances decrease from 400 (FedAvg) to 346–361 with RL. The distance curves (Figure 6) further suggest improved stability: lower average pairwise distances indicate reduced disagreement across client models, which is particularly valuable under stronger heterogeneity.

4.3.3 Summary

Across Figures 5–7 and Tables 2, RL-based adaptation is most beneficial under stronger non-IID data ($\alpha = 1$). Compared to fixed FedAvg-style policies, the learned controllers reduce compute and communication budget violations while maintaining competitive (often improved) accuracy. Adaptive layer sharing additionally improves training stability under heterogeneity, as reflected by reduced model divergence.

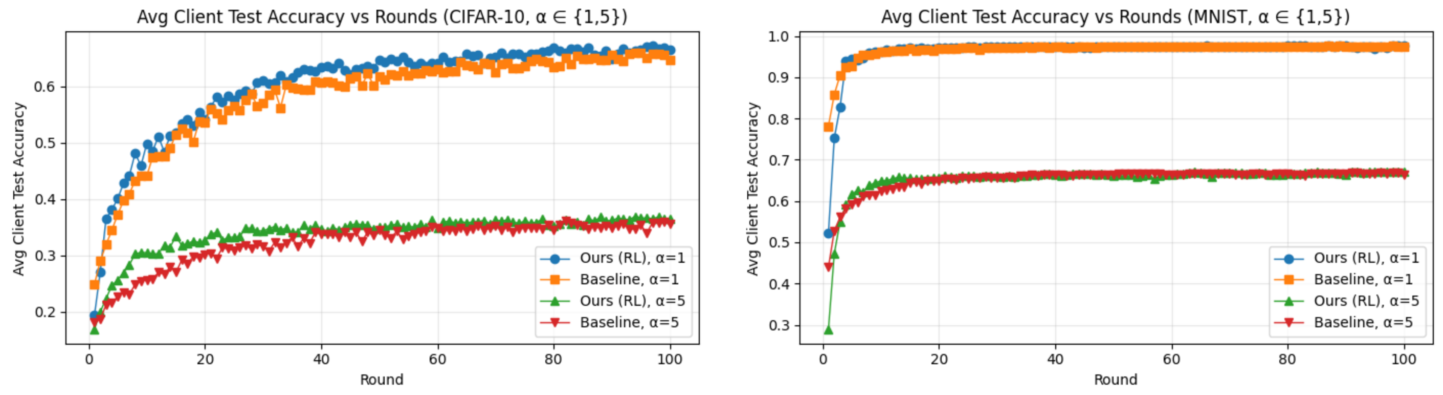
5 Conclusion

This report presented a reinforcement learning based personalized federated learning framework that makes per-



(a) CIFAR-10: Global test accuracy over communication rounds. (b) MNIST: Global test accuracy over communication rounds.

Figure 6: Comparison of our RL-based client layer selection policy and standard FedAvg under two Dirichlet non-IID settings ($\alpha = 1$ and $\alpha = 5$).



(a) CIFAR-10: Global test accuracy over communication rounds. (b) MNIST: Global test accuracy over communication rounds.

Figure 7: Client-averaged global test accuracy over communication rounds for our combined RL controller (training intensity plus shared layer selection) compared to the baseline, under two Dirichlet non-IID settings ($\alpha \in \{1, 5\}$).

client training decisions to better accommodate heterogeneous clients with non-IID data and different resource preferences. We studied three directions: (i) an RL agent for guiding training intensity control, (ii) an RL agent for guiding shared layer selection, and (iii) a single agent that jointly controls both training intensity and shared layer selection.

Across CIFAR-10 and MNIST under Dirichlet non-IID settings ($\alpha \in \{1, 5\}$), the training intensity agent achieved slightly higher client-averaged global test accuracy than the baseline while consistently reducing compute-budget exceedances. In the shared layer selection setting, the RL agent maintained competitive accuracy while reducing communication-budget exceedances compared to the FedAvg baseline, demonstrating that communication can be adapted per client without sacrificing performance. When combining training intensity control with layer selection in a single agent, we preserved or improved accuracy and further improved constraint compliance, reducing both compute and communication budget exceedances overall. These results indicate that round-local, context-aware decisions learned via a lightweight RL formulation can provide a stronger accuracy–efficiency trade-off than fixed policies in personalized federated learning.

6 Future Work

There are several promising directions to extend this work. First, we plan to adopt multi-objective or constrained RL formulations to more explicitly balance accuracy, compute, communication, and fairness across clients, and to reduce sensitivity to reward-weight tuning. Second, we will evaluate the learned policies under more realistic system heterogeneity, including client dropouts, stragglers, time-varying bandwidth, and energy constraints. Third, we aim to generalize beyond selecting only the number of shared layers by learning *which* layers or modules to share (e.g., selective blocks, adapters, or low-rank updates), potentially combined with communication compression such as quantization or sparsification. Finally, we will study robustness and privacy aspects, including the impact of differential privacy noise and resilience to adversarial clients, and provide deeper convergence and stability analysis for the proposed adaptive scheduling strategies.

References

[1] Yanfei Lin, Haiyi Wang, Weichen Li, and Jun Shen. Federated learning with hyper-network—a case study on whole slide image analysis. *Scientific Reports*, 13(1): 1724, 2023.

[2] TV Nguyen, MA Dakka, SM Diakiw, MD VerMilyea, M Perugini, JMM Hall, and D Perugini. A novel decentralized federated learning approach to train on globally distributed, poor quality, and protected private medical data. *Scientific Reports*, 12(1):8888, 2022.

[3] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE transactions on neural networks and learning systems*, 34(12): 9587–9603, 2022.

[4] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021.

[5] Benyuan Sun, Hongxing Huo, Yi Yang, and Bo Bai. Partialfed: Cross-domain personalized federated learning via partial initialization. *Advances in Neural Information Processing Systems*, 34:23309–23320, 2021.

[6] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in neural information processing systems*, 33: 21394–21405, 2020.

[7] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[8] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE conference on computer communications*, pages 1698–1707. IEEE, 2020.

[9] Hangjia Zhang, Zhijun Xie, Roozbeh Zarei, Tao Wu, and Kewei Chen. Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach. *IEEE Access*, 9:98423–98432, 2021.

[10] Gaith Rjoub, Omar Abdel Wahab, Jamal Bentahar, Robin Cohen, and Ahmed Saleh Bataineh. Trust-augmented deep reinforcement learning for federated learning client selection. *Information Systems Frontiers*, 26:1261–1278, 2024.

[11] Tingting Wu, Xiao Li, Pengpei Gao, Wei Yu, Lun Xin, and Manxue Guo. Resource-aware personalized federated learning based on reinforcement learning. *IEEE Communications Letters*, 2024.

[12] Hongwei Yang, Juncheng Li, Meng Hao, Weizhe Zhang, Hui He, and Arun Kumar Sangaiah. An efficient personalized federated learning approach in heterogeneous environments: a reinforcement learning perspective. *Scientific Reports*, 14(1):28877, 2024.

[13] Xi Chen, Qin Li, Haibin Cai, and Ting Wang. Heterogeneity-aware personalized federated learning via adaptive dual-agent reinforcement learning. *arXiv preprint arXiv:2501.16966*, 2025.

[14] Sai Qian Zhang, Jieyu Lin, and Qi Zhang. A multi-agent reinforcement learning approach for efficient client selection in federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 9091–9099, 2022.

[15] Mohammed Amir Messioud, Abdelhamid Malki, and Samir Ouchani. Deep reinforcement learning for client selection and resource allocation in federated learning: A comprehensive survey. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 536–546. Springer, 2025.

[16] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[17] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.