Ranking-based Client Imitation Selection for Efficient Federated Learning

Chunlin Tian [*] [1] **Zhan Shi** [*] [2] **Xinpeng Qin** [3] **Li Li** [1] **Chengzhong Xu** [1]

Abstract

Federated Learning (FL) enables multiple devices to collaboratively train a shared model while ensuring data privacy. The selection of participating devices in each training round critically affects both the model performance and training efficiency, especially given the vast heterogeneity in training capabilities and data distribution across devices. To deal with these challenges, we introduce a novel device selection solution called FedRank, which is based on an end-to-end, ranking-based model that is pre-trained by imitation learning against state-of-the-art analytical approaches. It not only considers data and system heterogeneity at runtime but also adaptively and efficiently chooses the most suitable clients for model training. Specifically, FedRank views client selection in FL as a ranking problem and employs a pairwise training strategy for the smart selection process. Additionally, an imitation learning-based approach is designed to counteract the cold-start issues often seen in state-of-the-art learning-based approaches. Experimental results reveal that FedRank boosts model accuracy by 5.2% to 56.9%, accelerates the training convergence up to 2.01× and saves the energy consumption up to 40.1%.

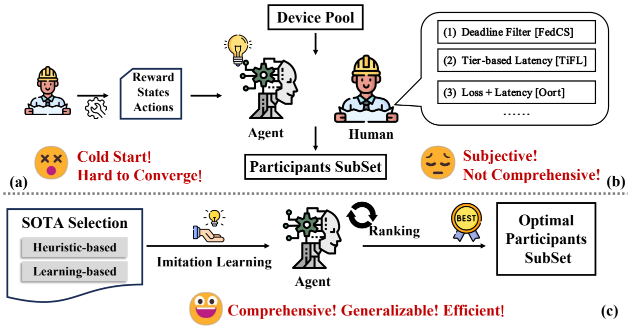


Figure 1. Illustration of FedRank. (a) Learning-based approaches: tackle device selection with data-driven processes that holistically trade-off demands. (b) Heuristic-based approaches: hand-developed heuristics perform well in the specific, straightforward deployment configurations they were designed for. (c) FedRank: utilizes imitation learning and a ranking approach to optimize device selection.

1. Introduction

Federated learning (FL) is a new learning paradigm that enables multiple devices to collectively train a global machine learning model while preserving data privacy (McMahan et al., 2017). However, the practical deployment of FL on diverse and resource-limited mobile devices poses several challenges, as illustrated in Figure 2. First, the disparity in training and communication capabilities of these devices, coupled with their unpredictable runtime variations, can result in stragglers. Consequently, the overall training throughput is often restricted by these low-end clients, leading to significant delays in the overall training process. Second, the data heterogeneity can influence the convergence and overall performance of the global model. In addition, computing-intensive local training and round-to-round communication with the central server can lead to substantial energy consumption, thereby affecting the battery lifespan of mobile devices.

To surmount the aforementioned barriers, strategic selection of devices for each training round becomes pivotal to both training efficiency and model performance, as depicted in Figure 1. While these heuristics excel in the specific, straightforward deployment scenarios they are designed for, they only cater to a limited subset of all potential deployment situations. As a result, their performance can be suboptimal on devices subjected to larger-scale deployments with greater diversity and complexity. Adapting to unfamiliar and intricate scenarios frequently necessitates profound domain expertise and extensive tuning. Conversely, certain studies (Zhan et al., 2020; Wang et al., 2020a; Zhang et al., 2022; Tian et al., 2023) explore the use of reinforcement learning (RL) techniques to address the device selection problem in a manner that holistically balances requirements using data-driven methods. Yet, real-world deployment sce-

[*] Equal contribution [1] University of Macau [2] University of Texas at Austin [3] University of Electronic Science and Technology of China. Correspondence to: Li Li <llili@um.edu.mo>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

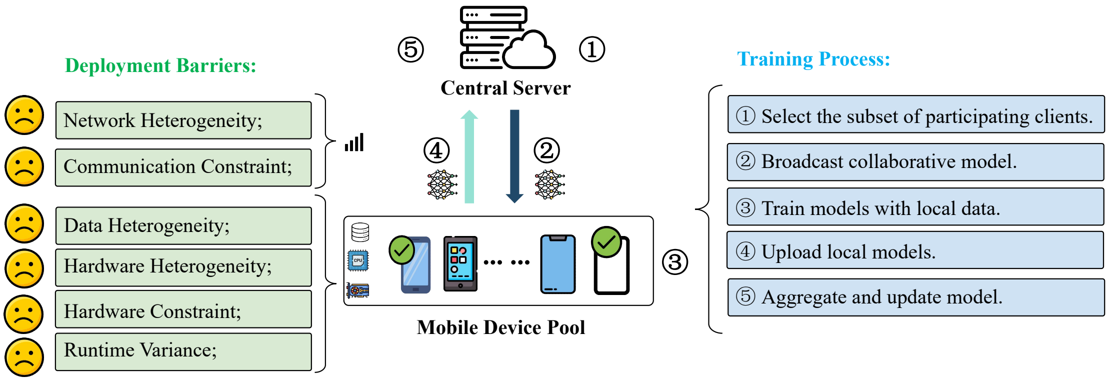


Figure 2. Workflow and real-world deployment barriers of Federated Learning.

narios feature a plethora of devices that constantly evolve. Such endeavors prove sample-inefficient and are plagued by cold-start problem.

This paper presents FedRank, a novel methodology for automated device selection in FL. We augment the current state-of-the-art in two pivotal ways. First, prior work typically assess each candidate device *in isolation* and predict their potential value score of devices when selected. While this approach aims to accurately predict the absolute artificial score, it is not trained directly to determine which device outperforms another—the essence of the problem. In response, *we draw parallels between device selection and the learning to rank (LTR) paradigm* (Liu et al., 2009; Qin et al., 2010; Casalegno, 2022). Our objective is to mold FedRank into a recommendation system, guiding the FL training agent toward the most impactful devices. To realize this vision, the recommendation system is devised to rank devices based on their significance, selecting the top-K. Embracing this perspective, we transition from a pointwise loss framework to a pairwise one (Joachims, 2002; Li, 2011), yielding pronounced advantages. Second, we identify that the *cold-start* issue has impeded the real-world utility of RL-based solutions, resulting in performance that is not as robust as that of analytical-based methodologies. To counteract this challenge, *we introduce a pre-training regimen that leverages a state-of-the-art analytical model through imitation learning (IL)* (Ross & Bagnell, 2014; Sun et al., 2017). Subsequently, we refine FedRank by further training with real-world interactions, enabling it to outstrip the analytical model's performance. In particular, this paper makes the following contributions:

- This paper casts device selection in federated learning as a ranking challenge, underscoring the superiority of pairwise training compared to previous methodologies.
- This paper proposes an offline pre-training scheme against state-of-the-art analytical solutions using imitation learning. This strategy eliminates the cold-start dilemma that has traditionally hindered the efficacy of learning-based device selection techniques.
- Building on these strengths, our proposed method, FedRank, adeptly synchronizes model performance, convergence, and energy efficiency amidst a dynamic and varied training environment. The efficacy of FedRank is substantiated through comprehensive experiments.

2. Background and Related Work

2.1. Federated Learning

Figure 2 provides a visualization of a typical federated learning framework, highlighting challenges that arise during real-world implementations. At the beginning of each training cycle, a central server selects a subset K of devices from the available pool N. It then distributes the initialized collaborative models to the chosen devices. Upon receiving the model, these selected devices conduct local training with their own data. Post-training, these devices transmit model updates back to the central server. The server then combines these updates, refining the overarching collaborative model. This cyclical process persists until the model either converges to a desired state or attains the target accuracy. Nonetheless, the variability in resources—including hardware specifications, data volume, and communication capacities—combined with stochastic runtime discrepancies can introduce inefficiencies, potentially compromising both training speed and model accuracy.

To surmount these challenges, selection of adept training devices becomes pivotal. Prior solutions have gravitated towards two predominant strategies: heuristic-based and learning-based approaches.

Heuristic-based selection. Traditional methods for device selection predominantly rely on heuristics rooted in isolated considerations such as data heterogeneity (Cho et al., 2020;

Balakrishnan et al., 2022), distinct training processes (Diao et al., 2021; Tam et al., 2023; Ma et al., 2024), and energy efficiency (Li et al., 2019). Advanced policies (Lai et al., 2021; Tian et al., 2022; Ning et al., 2024), have secured state-of-the-art results by employing analytical strategies that comprehensively address the multifaceted nature of device selection. Nevertheless, heuristic-based methodologies inherently risk escalating into NP-hard problems, especially with the expansive scale of FL deployments. Adapting these heuristics to unseen scenarios often demands a confluence of domain-specific expertise and extensive tuning.

Learning-based selection. Several contemporary approaches have pivoted towards devising policies for device selection through learning mechanisms. Specifically, AutoFL (Kim & Wu, 2021), Favor (Wang et al., 2020a) and FedMarl (Zhang et al., 2022) employ reinforcement learning methodologies (Sutton & Barto, 2018) to formulate decisions as Markov decision processes (MDPs). While AutoFL leverages a Q-table, Favor adopts Q-learning, and FedMarl utilizes Multi-Agent Reinforcement Learning (MARL) to achieve its objectives. In theory, these approaches are designed to select devices adeptly, however, uniformly demonstrating a tendency towards sample inefficiency are susceptible to the cold-start conundrum. This limitation often results in subpar selections during the initial phases. Consequently, the practical applicability of these methods in real-world scenarios remains circumscribed.

2.2. Imitation Learning & Learn to Rank

All of the above approaches (heuristic-based and learning-based) struggle in diverse and heterogeneous real-world deployment environments. A question is how they can complement each other in order to select a qualitatively more effective device?

Imitation learning. As a result, we introduce imitation learning (Ross & Bagnell, 2014; Sun et al., 2017) to utilize well-known heuristics as conditions (expert policy) to warm-up RL (supervised). Imitation learning is instructed by the expert's demonstration, assuming its optimal settings to learn a policy, so the current policy resembles the expert one. In the IL environment, there are two peculiarities: One is that the analytical policy can be queried at any training state, which is more efficient than an expensive expert during early stages of RL. The ability to probe the expert arbitrarily allows us to avoid the performance gap with the straggler. The other is that the distribution over actions of the analytical policy is made available, enabling more sophisticated loss functions. Previous work also investigates settings with these two properties, albeit in different domains. Sabour et al. (2018) shows that an approximate oracle can be computed in some natural language sequence generation tasks. Liu et al. (2020) learns to imitate Belady's, an oracle cache replacement policy.

Learn to rank. For an additional dimension, traditional works typically evaluated each candidate device in isolation and predicted the potential value score of the device if it were to be selected. While this pointwise approach aims to accurately predict the absolute artificial score, it is not directly trained to determine which device is superior to another, resulting in a loss function that may overemphasize non-critical devices with low performance. Learn to Rank (Liu et al., 2009; Qin et al., 2010; Casalegno, 2022; Ning et al., 2022) is the application of machine learning techniques for the creation of ranking models for information retrieval systems. Ranking models typically work by predicting a relevance score $s = f(x)$ for each input $x_i = (q, d)$ where q is a query and d is a document. Once the relevance of each document is known, it can be sorted (i.e., ranked) according to these scores to find contents of interest with respect to a query. Therefore, in this paper, we introduce pairwise (Joachims, 2002; Li, 2011) to enable the selected devices to score significantly higher than the remaining devices, which is computed as the sum of the loss terms defined on each pair of selected devices d_i, d_j, for $i, j = 1...n$. The goal of training the model is to predict whether the ground truth $y_i > y_j$ or not, which of two devices is more relevant.

3. FedRank: Framework Design

In this section, we introduce the design of FedRank. Specifically, we first cast the problem of client selection in FL as imitation learning through framing it as learning a policy within an episodic Markov decision process. After that, we describe the integration of imitation learning as a countermeasure to the cold-start issue. Finally, we introduce the learn-to-rank methodology to effectively augment the training performance.

3.1. Casting Device Selection as Imitation Learning

This serves as a foundation to outline the optimization space for FedRank that devices depend on a simple two-layer multi-layer perceptron (MLP) at the central server to perform the participation decision-making. The imitation learning phase aims to extract device selection actions A based on current states S, which are trained with the Value Decomposition Network (VDN) (Sunehag et al., 2017) to maximize the reward R.

STATE: Specifically, the state at the t-th timestep $s_t = (s_t^T, s_t^E, s_t^D) \in S$ consists of three components, where:

- $s_t^T = (T^t_{comp,i}, T^t_{comm,i})$ is the local training latency, consisting of training latency $T^t_{comp,i}$, communication latency $T^t_{comm,i}$, to capture system heterogeneity.

- $s_t^E = (E_{comp,i}^t, E_{comm,i}^t)$ is the energy cost of local training, consisting of computation energy cost $E_{comp,i}^t$ and communication energy cost $E_{comm,i}^t$, to reflect system heterogeneity.
- $s_t^D = (L_i^t, D_i^t)$ describes the data heterogeneity, where L_i is the training loss of device i and D_i is its data size.

How to obtain device state cost-effectively is the critical challenge. Existing approaches (Cho et al., 2020; Wang et al., 2020a; Lai et al., 2021; Tian et al., 2022; Wu et al., 2024) typically retrieve states that first force devices to complete all their local training along with updating the local DNN weights, resulting in significant processing latency. To address this issue, we introduce a "*early exit*" scheme that obtains the device states after the first epoch of local training (called "probing") processes at each device. Probing training is performed as an intermediate result with no additional computational overhead. In addition, sending the probing training results to the central server avoids introducing extra processing latency and communication costs, since the probing states (scalars) are tiny compared to the DNN model. On the contrary, this scheme will reduce the overall processing latency and communication overhead. This is for the reason that after the outlier devices exit, only a subset of devices need to complete their local training and send their weight updates in each training round.

ACTION: The action set A_t are N binary indicators representing whether each of the N devices available at timestamp t should be involved in ($a_i^t = 1$) or excluded from the incoming training round ($a_i^t = 0$).

REWARD: To characterize the primary optimization axes, we define three rewards: R_{acc}, R_T, and R_E denote the test accuracy of the global model, the processing latency and the total energy consumption of each round, respectively. Collectively, these rewards strike a balance among model performance, training efficiency, and energy-friendly during the training round t. They are denoted as:

$$R^t = \Delta R_{acc}^t \times (\frac{T}{R_T^t})^{\mathbf{1}(T<R_T^t)\times\alpha} \times (\frac{E}{R_E^t})^{\mathbf{1}(E<R_E^t)\times\beta}$$

where:

$$R_T^t = T_{prob} + \max_{i\in K}\{[T_{comm,i}^t + T_{comp,i}^t * (l_{ep} - 1)] * a_i^t\}$$
$$R_E^t = E_{prob} + \sum_K\{[E_{comm,i}^t + E_{comp,i}^t * (l_{ep} - 1)] * a_i^t\} \tag{1}$$

Where T and E are the developer-preferred duration and energy budget of the devices, respectively. l_{ep} is the number of local training epochs. T_{prob} and E_{prob} represent the latency and energy cost of early exit from the first local probing epoch, respectively. $\mathbf{1}(x)$ is an indicator function that takes the value 1 if x is true and 0 otherwise. In this way, the utility of those clients that may be the bottleneck of the desired speed and energy cost of the current round is penalized by a developer-specified factor α and β. However, we do not reward the non-straggler and non-overloader clients because their completions do not affect the effectiveness of the training round.

Despite the aforementioned optimization function setup employed in this paper, FedRank is highly flexible and configurable, allowing users to tailor the setup to diverse exploration objectives. Our objective is to conduct the optimal participant selection, which entails maximizing model performance in terms of accuracy and optimizing training efficiency in terms of both latency and energy-efficiency.

3.2. Offline Pre-Train with Imitation Learning

Unlike prior RL solutions that are trained entirely on-the-fly (Zhang et al., 2022; Wang et al., 2020a; Zhan et al., 2020), we pretrain the Q-Network in an offline manner using Behavioral cloning, a common approach in Imitation Learning, by imitating actions taken by an expert. Specifically, FedRank utilizes a three-layer Q-Network to approximate the Q-function for device selection, expressed as $Q_i^\theta(s, a) = \pi[R_t|s_i^t = s, a_i^t = a]$, where t represents the current round, and the states s_i^t, which is obtained by profiling the i-th device, function as the input to the network. Once the Q-values are computed, FedRank selects the devices with top-K Q-value and the corresponding actions are set to $a_i^t = 1$ while the others remain $a_i^t = 0$. $Q(\mathbf{s}_t, \mathbf{a}_t) = \sum Q_i^\theta(s_i^t, a_i^t)$, where $\mathbf{s}_t = \{s_i^t\}$ and $\mathbf{a}_t = \{a_i^t\}$ are aggregated from all devices.

Algorithm 1 summarizes the offline pre-train algorithm for FedRank policy π_θ. The high-level scheme is to observe a set of states B and then update the parameters θ to make the same device decision as the optimal state-of-the-art analytical policies π^* as the experts for each state $s \in B$ via the loss function $L_\theta(s, \pi^*)$. Specifically, we perform the analytical policies (Harmony (Tian et al., 2022), Oort (Lai et al., 2021), FedMarl (Zhang et al., 2022)) with a given set of mobile devices and obtain episodes of state-action pairs to form an expert demonstration B (lines 3-5). Given the states, we train FedRank with truncated backpropagation through time (lines 6-9). We first sample batches of states with the demonstration B to initialize selection policy π_θ. Then, we compute the the loss $\mathcal{L} = \sum_{i=k} L_\theta(\mathbf{s}_t, \pi_*)$ and update the policy parameters θ based on the loss L. The pretraining encourages the learned selection policy $\pi_\theta(a_t|s_t)$ to make decisions that approximate the analytical policy π^*.

3.3. Online Learning Process

In the deployment setting, we apply the pre-trained model to the online FL training optimization process. Meanwhile, we also adopt the target Q-network and the predict Q-network

Algorithm 1 FedRank Imitation Learning Algorithm.

```
Initialize policy $\pi_\theta$
for step $= 0$ to $K$ do
    if step $\equiv 0$ then
        Collect dataset of state-action pairs to form a visited states $B = \{\mathbf{s}_t\}_{t=0}^{T}$ by perform the analytical policies on a given set of mobile devices.
    end if
    Sample states $\mathbf{s}_t$ from $B$
    Warm up policy $\pi_\theta$ on sampled states.
    Compute loss $\mathcal{L} = \sum_{i=k} L_\theta(\mathbf{s}_t, \pi_*)$
    Update policy parameters $\theta$ based on loss $\mathcal{L}$
end for
```

to enhance the adaptability of the post-IL network to new environments. Therefore, the Q-Network is trained recursively with minimum loss:

$$L = \pi_{\mathbf{s}^t,\mathbf{a},r,\mathbf{s}^{t+1}}[r_t + \gamma * \sum_i Q_i^{\theta'}(\mathbf{s}_i^{t+1}, a) - Q_i^{\theta}(\mathbf{s}_i^t, \mathbf{a})] \quad (2)$$

where $\theta^{'}$ denotes the target network parameters that are periodically copied from θ throughout the entire training phase. When all devices have completed their DNN inference, the scheduler receives a global reward r_t and proceeds to the next state s_i^{t+1}. The Profiler Cache is used to store the tuple of transitions $< s_i^t, a_i^t, s_i^{t+1}, r_t >$ for each node i to train the DNN.

3.4. Pairwise Ranking Loss

The ranking of the Q-values rather than the absolute values determines the selection of devices. The Q-Network trained to preserve orders among Q-values fits the problem better than the Q-Network trained to predict absolute Q-values. Thus, we adopt the pairwise loss (Joachims, 2002; Li, 2011) defined on each pair of selected devices d_i, d_j, for $i, j = 1...n$. Specifically, in pairwise training, each pair of network outputs are mapped to a binary indicator, which is trained to minimize its distance to $y_{i,j}$ (=1 if $y_i > y_j$, 0 otherwise). We then redefine the RL rewards with the pairwise loss in Eq.(2) to prioritize the relative orders over the absolute values of rewards. We map the output Q-value (predict Q and target $\bar{Q}$) to probabilities using a logistic function as follows:

$$\begin{aligned} P_{i,j} &= \sigma[Q(s_i, a_i) - Q(s_j, a_j)] \\ \overline{P_{i,j}} &= \sigma[\bar{Q}(s_i, a_i) - \bar{Q}(s_j, a_j)] \end{aligned} \quad (3)$$

Subsequently, we employ RankNet (Burges et al., 2005) as a pairwise method, which uses a Binary Cross Entropy (BCE) loss to represent the pairwise loss of devices i and j as in Equation (4). We can then obtain the average pairwise loss:

$$L_{Rank} = -\sum_{i,j=1}^{n} \overline{P_{i,j}} \log P_{i,j} + \left(1 - \overline{P_{i,j}}\right) \log\left(1 - P_{i,j}\right) \quad (4)$$

The new joint loss function of the RL DNN model can be defined as:

$$L = \overline{L_{RL}} + \epsilon * \overline{L_{Rank}} \quad (5)$$

4. Experiments

4.1. Experimental Setup

Infrastructure. We evaluated the effectiveness of FedRank using a hybrid testbed with both simulation and off-the-shelf mobile devices which effectively emulates both data and system heterogeneity in real-world cases. Specifically, we first built a simulator following the server/client architecture based on PyTorch (Paszke et al., 2019), in which different processes were created to emulate the central server and the participating devices with heterogeneous data distribution. Monsoon Power Monitor (Monsoon, 2023) was utilized to monitor energy consumption during the training process. At the same time, the user interaction traces (Shepard et al., 2011) were further integrated to emulate the concurrently running applications that impact the training capability at runtime.

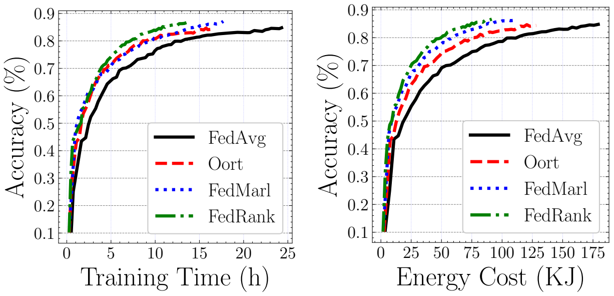
Models and datasets. The following representative models and datasets were utilized for evaluation. For the datasets, in-domain (ID) refers to the datasets consistent with imitation learning, and out-of-domain (OOD) indicates the invisible datasets. Specifically, MNIST (Lecun et al., 1998) on LeNet5 (LeCun et al., 1998) was adopted as the ID dataset. While for OOD datasets, ResNet18 (He et al., 2016) on CIFAR10 (Krizhevsky et al., 2009), VGG16 (Simonyan., 2014) on CINIC10 (Darlow et al., 2018), ShuffleNet (Zhang et al., 2018) on TinyImageNet (Deng et al., 2009) for image classification were utilized. Furthermore, Dirichlet distribution $p_k \sim Dir_N(\sigma)$ (Wang et al., 2020b) was utilized to simulate the non-IID data distribution across different devices.

Baselines. We compared FedRank with three types of representative device selection approaches, including **A. Random selection:** (1) *FedAvg* (McMahan et al., 2017) a vanilla framework for FL without any operation. (2) *FedProx* (Li et al., 2021) dynamically tunes the randomly selected local device iterations utilizing training loss to keep the system robustness. **B. Heuristic-based selection:** (3) *AFL* (Goetz et al., 2019) selects each round of participating devices with a probability conditioned on the current model, as well as the data on the client, to maximize efficiency. (4) *TiFL* (Chai et al., 2020) selects devices with similar response latency for each round to reduce system heterogeneity. (5) *Oort* (Lai

Table 1. Model performance of different selection approaches about test accuracy, energy cost per round on average, and training speed per round on average. FedRank achieves the best performance across all the datasets. [1] Set $\sigma = 0.01$ for MNIST, $\sigma = 0.1$ for others to emulate Non-IID. [2] Training round that target accuracy (99%) is achieved.

Dataset &Model		ID			OOD								
		MNIST-LeNet5			CIFAR10-Resnet18			CINIC10-VGG16			TinyImageNet-ShuffleNet		
		Acc (%) ↑	Energy ↓	Speed ↑	Acc (%) ↑	Energy ↓	Speed ↑	Acc (%) ↑	Energy ↓	Speed ↑	Acc (%) ↑	Energy ↓	Speed ↑
IID	FedAvg	98.84	100%	1×	84.78	100%	1×	64.78	100%	1×	34.62	100%	1×
	FedProx	$99^{2}_{(35)}$	91.7%	1.03×	85.11	91.7%	1.01×	64.51	95.2%	1.05×	35.38	99.0%	1.04×
	AFL	98.61	75.7%	1.10×	83.33	78.7%	1.08×	65.86	86.2%	1.07×	36.77	93.5%	1.15×
	TiFL	98.91	71.9%	1.19×	84.96	67.6%	1.11×	67.32	76.3%	1.35×	38.8	70.9%	1.33×
	Oort	$99_{(49)}$	55.8%	1.26×	85.69	55.8%	1.40×	67.07	69.4%	1.62×	40.57	76.3%	1.51×
	Favor	98.87	84.0%	1.03×	85.98	81.3%	1.10×	65.47	97.1%	1.15×	37.62	99.0%	1.24×
	FedMarl	98.82	48.1%	1.07×	86.37	53.2%	1.30×	66.74	68.0%	1.86×	39.48	75.2%	1.67×
	FedRank	**99**$_{(32)}$	**40.1%**	**1.56**×	**87.69**	**44.4%**	**1.67**×	**68.58**	**52.9%**	**2.01**×	**43.54**	**63.3%**	**1.81**×
Non-IID	FedAvg	36.70	100%	1×	42.02	100%	1×	37.40	100%	1×	23.82	100%	1×
	FedProx	67.67	95.2%	1.04×	52.40	101%	1.04×	39.59	99.3%	1.05×	24.72	98.7%	1.03×
	AFL	89.60	90.1%	1.13×	51.81	87.7%	1.10×	41.03	93.0%	1.17×	25.90	93.4%	1.08×
	TiFL	91.21	76.3%	1.21×	56.39	66.7%	1.20×	42.16	74.9%	1.34×	25.85	73.6%	1.27×
	Oort	92.47	71.4%	1.36×	53.51	61.7%	1.38×	43.57	68.3%	1.46×	26.62	71.5%	1.51×
	Favor	85.48	90.1%	1.08×	51.34	93.4%	1.16×	39.26	96.8%	1.19×	25.34	93.5%	1.20×
	FedMarl	90.13	65.8%	1.16×	56.73	60.2%	1.35×	44.16	67.4%	1.67×	26.31	68.8%	1.57×
	FedRank	**93.67**	**47.4%**	**1.48**×	**59.11**	**50.8%**	**1.78**×	**47.07**	**55.2%**	**1.75**×	**30.04**	**67.4%**	**1.83**×

et al., 2021) selects the optimal participating devices based on the user-defined utility value that combines training loss and latency. **C. Learning-based selection:** (6) *Favor* (Wang et al., 2020a) adopts accuracy as an indicator to learn local weight selection to reduce Non-IID effects while speeding up convergence. (7) *FedMarl* (Zhang et al., 2022) employs multi-agent reinforcement learning for device selection. It randomly chooses devices as RL agents each round, optimizing the final participant selection with accuracy, training latency, and communication cost. Specifically, we set the reward hyperparameters $\alpha = \beta = 2$ in Eq. 1. This is for the reason that FedRank jointly considers energy consumption and training time in each round in the reward function. We set the device pool $N = 100$, and $K = 10$ devices were selected to participate according to various selection policies, with $r = 50$ training rounds and $l_{ep} = 5$ local training epochs per round. (More details are given in Appendix A).



(a) ToA: time to accuracy

(b) EoA: energy to accuracy

Figure 3. Efficiency comparison of various schemes to train the ResNet18 model on CIFAR10 (IID). We compared the FedRank with the SOTA in the three baselines, respectively.

4.2. Overall Performance

Model performance. Table 1 compares the model performance of FedRank with the baselines. FedRank achieves significantly higher accuracy and converges faster than the others, 56.9% (Non-IID) on the ID dataset. For the OOD datasets, FedRank improves the test accuracy up to 8.92% on IID and 17.08% on Non-IID over all datasets, while 5.2% on IID and 11.0% on Non-IID averaged. There are two possible reasons 1) the robust and effective groundwork laid by high-quality SOTA selection approaches through offline IL sets a solid base for the development of discerning online RL selection. 2) the pairwise loss correction term accentuates the selection or non-selection of a device, thereby elevating variance. This adjustment facilitates the model's ability to more effectively discern and assimilate the distinct attributes of devices. Overall, this experiment demonstrates the practical and robust ability of FedRank to scale to complex workloads and application scenarios.

System efficiency. From Table 1, we also can see that FedRank effectively speeds up the training process with faster convergence up to 2.01× and achieves significant energy saving up to 40.1%. This is for the reason that FedRank jointly considers energy consumption and training time in each round in the reward function. In contrast, no baselines are SOTA in all cases, due to incomplete consideration. Oort emphasizes solely training duration as the system effectiveness metric, while FedMarl takes into account the energy costs associated with communication. However, it overlooks the intrinsic energy overheads of the training process itself, a critical oversight, especially for devices operating on battery power. We further conduct a comprehensive evaluation of the system efficiency of FedRank from two critical aspects: Time to Accuracy (ToA) and Energy to Accuracy (EoA). As depicted in Figure 3 (a), FedRank markedly accelerates the training phase, demonstrating rapid convergence paired with enhanced accuracy. Furthermore, Figure 3 (b) illustrates that FedRank also realizes considerable energy savings, underlining its efficiency. This can be attributed to FedRank thoughtfully integrates considerations of both latency and energy consumption during training. In comparison, Oort focuses exclusively on training time as the measure of system effectiveness. Conversely, while FedMarl accounts for the energy expenses related to communication, it neglects the inherent energy demands of the training phase, a significant lapse, particularly for battery-operated devices.

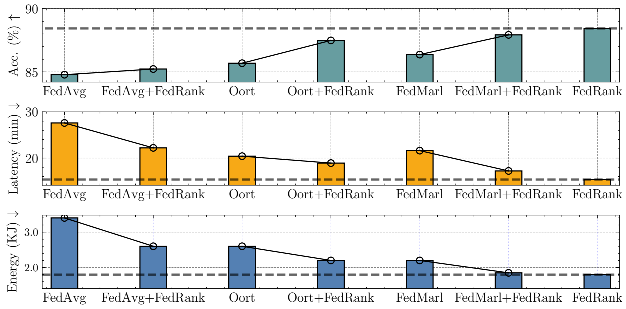


Figure 4. Generalization ability of FedRank to train the ResNet18 model on CIFAR10 (IID).

Generalization and robustness. An effective device selection policy must be able to suit unseen real-world deployments (e.g., datasets and training models), as there are large amounts of mobile devices in real-world cases. Moreover, as the states are highly diverse and time-evolving, encountering all of them during imitation learning is infeasible. In this experiment, we test FedRank's ability to generalize to unseen deployments. Table 1 represents the corresponding results. The model performance shows that with ID datasets (i.e., MNIST), FedRank improves the model convergence speed and model accuracy significantly. For OOD datasets, for the cases with simpler data and similar to domain datasets, it also shows significant performance improvement (CIFAR10). For more comprehensive datasets (CINIC10), the global training model in the baseline setting all perform inferiorly in heterogeneous real-world deployments. In contrast, FedRank maintains high accuracy even in more complicated and self-unseen scenarios, indicating that it can effectively generalize to unseen deployment environments during imitation learning.

4.3. Ablation Study

To explore the impact of the different modules of FedRank, we develop three model variants: (1) FedRank^{-I}, where offline imitation learning is ablated before FL. (2) FedRank^{-P}, where rank loss is ablated at DQN training. (3) FedRank^{-IP}, which directly employ RL to select devices.

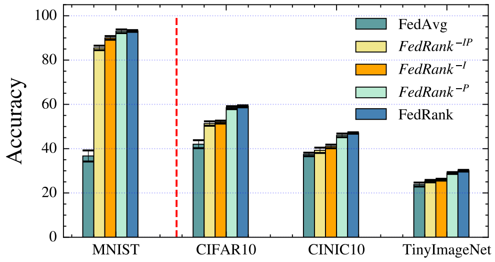
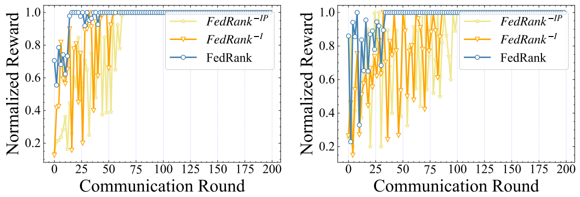


Figure 5. Ablation study explores the effects of imitation learning and pairwise loss on model performance across several datasets (MNIST, CIFAR10, CINIC, and TinyImageNet) under Non-IID settings. Model^{-I} (without imitation learning), Model^{-P} (without pairwise loss), and Model^{-IP} (lacking both features).

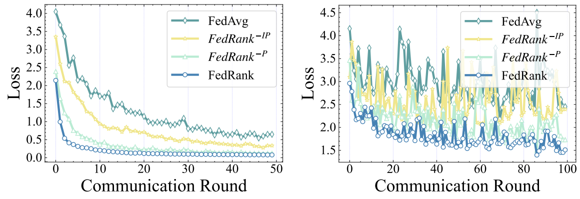


(a) MNIST-LeNet5, ID (b) VGG16-CIFAR10, OOD

Figure 6. Reward function analysis in the ablation study for imitation learning.

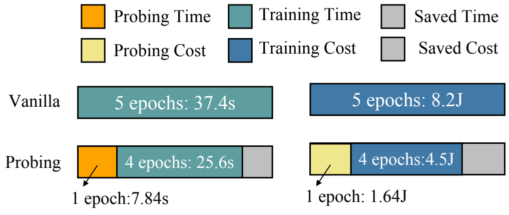
Effectiveness of imitation learning. Figure 5 shows that simply casting device selection as an RL problem with DQN neural architecture (FedRank^{-IP}) only produces a negligible gain as RL faces the cold-start issue. Therefore, our additions are required to achieve better model performance with *right* device selection. Meanwhile, it indicates that FedRank^{-I} only marginally increases efficiency in FL model training performance, while employing imitation learning can greatly enhance the accuracy by tackling the cold-start issue of the RL model in the early training round. We further present an analysis of the reward variation tendency of the RL DNN model. Figures 6 show that FedRank^{-IP} trained from scratch, the reward converges on average after about 70-100 aggregation rounds – more than 200 rounds are usually required for FL convergence. Consequently, using IL to pre-train FedRank leads to faster convergence due to the rich reservoir of knowledge.

Effectiveness of pairwise loss. In addition, as Figure 5 (d) shows, the ranking loss better optimizes model performance, as it more heavily penalizes non-critical devices with lower utility values, which impairs model performance. In addition, from a distillation perspective of loss, the probability of selecting a device is proportional to the reward value for performing the ranking, which suggests that ranking



(a) MNIST-LeNet5, ID (b) VGG16-CIFAR10, OOD

Figure 7. Training loss analysis in the ablation study of pairwise.



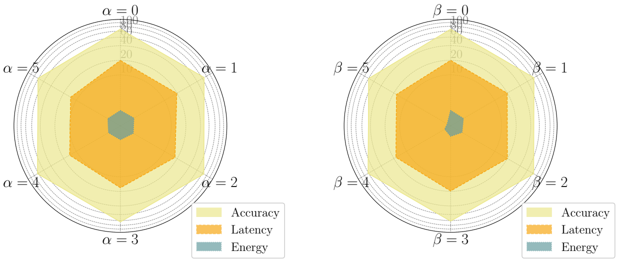
(a) Training Latency (b) Energy Cost

Figure 8. Average training latency and energy cost per round (5 local epochs) to train LeNet5 on MNIST. **Vanilla**: Full local training over 5 epochs to assess model bias. **Probing**: Clients perform the *1st* local epoch, then report the probing loss and training overhead information to the central server. Based on this, the server performs an early rejection and stops further 4-epoch training of clients with high bias and low performance.

loss provides greater supervision than the pointwise loss of FedRank^{-P}. Figure 7 compares the federated learning model training convergence loss. We can observe that rank loss speeds up the training process and accelerates the model convergence rate.

Generalization ability of FedRank. Figure 4 outlines when baselining the ResNet18 model via IID CIFAR10, the generalizability of FedRank in combination with different client selection methods and the effectiveness of using multiple selection methods for imitation learning. The adoption of a single selection method for imitation learning can significantly improve results over the corresponding baseline, highlighting the merits of continuous space paradigms in enhancing generalizability. Furthermore, the use of diverse, multiple selection methods for imitation learning (*i.e.*, the FedRank) outperforms the single strategy, emphasizing the value of diversity in imitation learning for comprehensive imitation.

Impact of penalty factors α and β of FedRank. FedRank uses two penalty factors to penalize the utility of high latency and energy devices in participant selection, whereby it adaptively prioritizes high system efficiency participants. Figure 9 shows that FedRank outperforms its counterparts across different α and β. Note that FedRank orchestrates its components to automatically navigate the best performance across parameters: larger α and β overemphasizing system efficiency drives the Pacer to relax the system constraint to admit clients with higher statistical efficiency. and vice versa. As such, FedRank achieves similar performance across all non-zero α and β.



(a) α (b) β

Figure 9. Penalty factors sensitivity of FedRank. α is the training latency penalty factor to avoid straggler, and β is the energy penalty factor to avoid high cost devices.

Impact of probing (early exit). Utilizing the Monsoon Power Monitor, we evaluate the training latency and energy consumption of the probing approach. Fig. 8 shows the training latency and energy consumption for one training round under the following two schemes: 1) Vanilla and 2) Probing. Specifically, Vanilla conducts full local training over 5 epochs to access model bias. While, with probing, the clients conduct the 1st epoch of local training, then report the probing loss and training overhead to the central server. Based on this, the server performs an early rejection and stops further local training of clients with high bias and low performance. In this case, the conventional approach consumes 37.4s of latency and 8.2J of energy in executing a local training round, effectively decreasing the latency by 10.6% and energy consumption by 25.2% by probing for early exit. We can see that probe training can effectively reduce resource and computational overhead and significantly improve efficiency. Meanwhile, probing data consisting only of scalars incurs negligible communication overhead compared to full model parameters.

5. Conclusion

Federated learning is making significant strides in secure environments. Real-world FL requires selecting the right participant subset, factoring in diverse use-cases, data, systems, and dynamic changes. Our work introduces a foundational, ranking-based device selection method for efficient FL, named FedRank. This method treats device selection as a ranking problem, using a pairwise training approach for intelligent selection, and continually identifying the best device group. To overcome initial challenges, FedRank employs an offline pre-training strategy, guided by advanced analytical solutions and imitation learning. The experimental results demonstrate that FedRank achieves a balance in FL between model performance and training efficiency.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgment

This paper is supported by the Science and Technology Development Fund (FDCT)) of Macau SAR (File no. 0081/2022/A2, 0123/2022/AFJ), MYRG-GRG2023-00211-IOTSC-UMDF, and the Start-up Research Grant of University of Macau (SRG2022-00010-IOTSC). Please ask Dr. Li Li (llili@um.edu.mo) and Dr. ChengZhong Xu (czxu@um.edu.mo) for correspondence.

References

Balakrishnan, R., Li, T., Zhou, T., Himayat, N., Smith, V., and Bilmes, J. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*, 2022.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96, 2005.

Casalegno, F. Learning to rank: A complete guide to ranking using machine learning. https://towardsdatascience.com, 2022.

Chai, Z., Ali, A., Zawad, S., Truex, S., Anwar, A., Baracaldo, N., Zhou, Y., Ludwig, H., Yan, F., and Cheng, Y. Tifl: A tier-based federated learning system. In *Proceedings of the 29th international symposium on high-performance parallel and distributed computing*, pp. 125–136, 2020.

Cho, Y. J., Wang, J., and Joshi, G. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

Darlow, L. N., Crowley, E. J., Antoniou, A., and Storkey, A. J. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Diao, E., Ding, J., and Tarokh, V. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021.

Goetz, J., Malik, K., Bui, D., Moon, S., Liu, H., and Kumar, A. Active federated learning. *arXiv preprint arXiv:1909.12641*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Joachims, T. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, 2002.

Kim, Y. G. and Wu, C.-J. Autofl: Enabling heterogeneity-aware energy efficient federated learning. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 183–198, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

Lai, F., Zhu, X., Madhyastha, H. V., and Chowdhury, M. Oort: Efficient federated learning via guided participant selection. In *OSDI*, pp. 19–35, 2021.

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

Lecun, Y., Cortes, C., and Burges, C. J. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

Li, H. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10): 1854–1862, 2011.

Li, L., Xiong, H., Guo, Z., Wang, J., and Xu, C.-Z. Smartpc: Hierarchical pace control in real-time federated learning system. In *2019 IEEE Real-Time Systems Symposium (RTSS)*, pp. 406–418. IEEE, 2019.

Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10713–10722, 2021.

Liu, E., Hashemi, M., Swersky, K., Ranganathan, P., and Ahn, J. An imitation learning approach for cache replacement. In *International Conference on Machine Learning*, pp. 6237–6247. PMLR, 2020.

Liu, T.-Y. et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3): 225–331, 2009.

Ma, J., Tian, C., Li, L., and Xu, C. Fedmg: A federated multi-global optimization framework for autonomous driving control. In *2024 IEEE/ACM 32st International Symposium on Quality of Service (IWQoS)*, pp. 1–10. IEEE, 2024.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Monsoon. Monsoon high voltage power monitor. https://www.msoon.com/online-store/High-Voltage-Power-Monitor-p90002590, 2023.

Ning, Z., Wang, P., Wang, P., Qiao, Z., Fan, W., Zhang, D., Du, Y., and Zhou, Y. Graph soft-contrastive learning via neighborhood ranking. *arXiv preprint arXiv:2209.13964*, 2022.

Ning, Z., Tian, C., Xiao, M., Fan, W., Wang, P., Li, L., Wang, P., and Zhou, Y. Fedgcs: A generative framework for efficient client selection in federated learning via gradient-based optimization. *arXiv preprint arXiv:2405.06312*, 2024.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Qin, T., Liu, T.-Y., Xu, J., and Li, H. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13:346–374, 2010.

Ross, S. and Bagnell, J. A. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Sabour, S., Chan, W., and Norouzi, M. Optimal completion distillation for sequence learning. *arXiv preprint arXiv:1810.01398*, 2018.

Shepard, C., Rahmati, A., Tossell, C., Zhong, L., and Kortum, P. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2011.

Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International conference on machine learning*, pp. 3309–3318. PMLR, 2017.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tam, K., Li, L., Han, B., Xu, C., and Fu, H. Federated noisy client learning. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2023. doi: 10.1109/TNNLS.2023.3336050.

Tian, C., Li, L., Shi, Z., Wang, J., and Xu, C. Harmony: Heterogeneity-aware hierarchical management for federated learning system. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 631–645. IEEE, 2022.

Tian, C., Shi, Z., and Li, L. Learn to select: Efficient cross-device federated learning via reinforcement learning. In Maughan, K., Liu, R., and Burns, T. F. (eds.), *The First Tiny Papers Track at ICLR 2023, Tiny Papers @ ICLR 2023, Kigali, Rwanda, May 5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=wecTsVkrjit.

Wang, H., Kaplan, Z., Niu, D., and Li, B. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1698–1707. IEEE, 2020a.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020b.

Wu, Y., Li, L., Tian, C., and Xu, C. Breaking the memory wall for heterogeneous federated learning with progressive training. *arXiv preprint arXiv:2404.13349*, 2024.

Zhan, Y., Li, P., and Guo, S. Experience-driven computational resource allocation of federated learning by deep reinforcement learning. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 234–243. IEEE, 2020.

Zhang, S. Q., Lin, J., and Zhang, Q. A multi-agent reinforcement learning approach for efficient client selection in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9091–9099, 2022.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on* *computer vision and pattern recognition*, pp. 6848–6856, 2018.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

A. Experiment Details

A.1. Dataset Details

For four computer vision datasets, we generate IID data splits by randomly assigning training examples to each client without replacement. For Non-IID splits, we simulate data heterogeneity by sampling label ratios from a Dirichlet distribution $p_k \sim Dir_N(\sigma)$ with the symmetric parameter σ. We set $\sigma = 0.01$ for MNIST, and $\sigma = 0.1$ for others to emulate Non-IID. For two natural language processing datasets, Shakespeare and Wikitext are naturally non-IID. We use LeNet5 (LeCun et al., 1998) on MNIST, ResNet18 (He et al., 2016) on CIFAR10, VGG16 (Simonyan., 2014) on CINIC10, ShuffleNet (Zhang et al., 2018) on TinyImageNet, LSTM (Hochreiter & Schmidhuber, 1997) on Shakespeare. For a fair comparison, baselines and FedRank will be compared under the same settings.

A.2. Baseline Details

We compare FedRank with three types of representative device selection approaches, including

- Random Selection:
  - **FedAvg** (McMahan et al., 2017), a vanilla framework for FL without any operation. In each round t, the server randomly selects a subset of available devices K_t from the total devices K. Each chosen device k trains the model on its local dataset using the current global model parameters w_t, resulting in updated local parameters w_{t+1}^k. The server aggregates these updated local parameters using the formula:

$$w_{t+1} = \frac{1}{K_t} \sum_{k=1}^{K_t} w_{t+1}^k \tag{6}$$

    This step effectively averages the local updates to form the new global model parameters.
  - **FedProx** (Li et al., 2021) improves FedAvg by handling heterogeneous data and systems, adding a proximal term to the local training objective to address non-IID data and system differences. In each round t, a subset of devices K_t is selected to train on their local data with the global model parameters w_t and a proximal term. The local update w_{t+1}^k is obtained by minimizing $L_k(w) + \frac{\mu}{2}\|w - w_t\|^2$, where $L_k(w)$ is the local loss, and μ adjusts the proximal term's influence. FedProx's main advantage is its ability to stabilize training across diverse data and system environments by ensuring local updates remain close to the global model.
- Heuristic-based Selection:
  - **AFL** (Goetz et al., 2019) uses a device selection method conditioned on the model and client data to enhance efficiency in each round t. This method focuses on devices likely to offer significant model improvements based on data diversity, model uncertainty, or past updates' impact. Post-training, devices submit their updates and informativeness measures (like loss or uncertainty) to the server. The server then combines these updates, potentially weighted by informativeness, to revise the global model to w_{t+1}, the following:

$$w_{t+1} = \sum_{k=1}^{K_t} \alpha_k w_{t+1}^k \tag{7}$$

    Here, α_k denotes the weight of the k-th device's contribution.
  - **TiFL** (Chai et al., 2020) groups devices by response latency to mitigate system heterogeneity, sorting them into tiers by capability, bandwidth, and data quality. Each tier has a Tier Server (TS) where selected devices contribute to local updates. These updates are first aggregated within tiers at the TS using:

$$w_{t+1}^{TS} = \frac{1}{K_t} \sum_{k=1}^{K_t} w_{t+1}^k \tag{8}$$

    Then, Tier Servers forward their updates to a central server for global aggregation, potentially weighted by tier attributes:

$$w_{t+1} = \sum_{i=1}^{T} \beta_i w_{t+1}^{TS_i} \tag{9}$$

    This balances individual contributions and overall data characteristics.

- **Oort** (Lai et al., 2021) optimizes device selection by integrating training loss and latency into a user-defined utility. To enhance efficiency, it's crucial to maximize the per-unit-time statistical utility. The utility of client i is defined as a product of her statistical utility and the global system utility, considering the duration of each training round, as shown in the equation:

$$\text{Util}(i) = \underbrace{|B_i| \sqrt{\frac{1}{|B_i|} \sum_{k \in B_i} \text{Loss}(k)^2}}_{\text{Statistical utility } U(i)} \times \underbrace{\left(\frac{T}{t_i}\right)^{\mathbf{1}(T<t_i)\times\alpha}}_{\text{Global sys utility}} \tag{10}$$

- Learning-based Selection:
  - **Favor** (Wang et al., 2020a) utilizes accuracy to determine local weight selection, mitigating non-IID impacts and enhancing convergence. It involves training a DRL agent via a double-deep Q-learning Network (DQN). Despite evident disparities in local model weights, which hold guiding data for device selection, the DQN agent's training aims to optimize the expected total discounted reward, represented as $R = \sum_{t=1}^{T} \gamma^{t-1} r_t = \sum_{t=1}^{T} \gamma^{t-1} \left(\Xi^{(\omega_t - \Omega)} - 1\right)$.
  - **FedMarl** (Zhang et al., 2022) applies multi-agent reinforcement learning to select devices for federated learning. It strategically selects RL agents each round to optimize accuracy, training latency, and communication cost. The reward r_t for each round t is given by:

$$r_t = w_1[U(Acc(t)) - U(Acc(t-1))] - w_2 H_t - w_3 B_t \tag{11}$$

where H_t, the processing latency, is:

$$H_t = \max_{1 \leq n \leq N} \left(H_{t,n}^{p}\right) + \max_{n:1 \leq n \leq N, a_n^t = 1} \left(H_{t,n}^{rest} + H_{t,n}^{u}\right) \tag{12}$$

In this, $\max_{1 \leq n \leq N} H_{t,n}^{p}$ signifies the maximum time for generating probing losses, used by MARL agents for client selection and model update. The time for client n to complete training and upload updates is $\max_{n:1 \leq n \leq N, a_n^t = 1} \left(H_{t,n}^{\text{rest}} + \ H_{t,n}^{u}\right)$ Here, $U(.)$ is a utility function ensuring even modest improvements in $Acc(t)$ are recognized towards the end of the learning process, and B_t represents the total communication cost.

B. Convergence Analysis

In this section we present a convergence analysis as a proof of the stability of FedRank, based on the works in (Zhao et al., 2018; Zhang et al., 2022). Given that each client $n \in N$ contains D_n local training data with underlying probability $P_n(.)$. Denote $\boldsymbol{X} \times \boldsymbol{Y}$ as the compact space and label space $[C]$, where class $[C] = \{1, ..., C\}$. While $\{\boldsymbol{x}, y_{\boldsymbol{x}}\}$ as the training data point and its ground truth label. Also, denote $f_m(\boldsymbol{x}, \boldsymbol{w})$ as the output probability for input $\boldsymbol{x}$ to take label m using DNN model f, where $\boldsymbol{w}$ is the weight of the neural network. On each device, local conventional SGD is conducted separately. At timestamp t on device $n \in N$, local performs: $\boldsymbol{w}_{n\_loc}^{t} = \boldsymbol{w}_{n\_loc}^{t-1} - \eta \nabla_{\boldsymbol{w}} \ell\left(\boldsymbol{w}_{n\_loc}^{t-1}\right)$. where $\ell(\boldsymbol{w}) = \sum_{i=1}^{C} p^{(n)}(y=i) \mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x}, \boldsymbol{w}_{n\_loc}^{t-1}\right)\right]$. Then, let $\boldsymbol{w}_{glb}^{t} = \sum_{n=1}^{N} \frac{D_n}{\sum_{n=1}^{N} D_n} \boldsymbol{w}_{n\_loc}^{t}$ denotes the trained DNN weights after t-th update by aggregating the local weight. Theorem 1 provides a bound on $\left|\boldsymbol{w}_{glb}^{t} - \boldsymbol{w}_{n\_loc}^{t}\right|$.

Theorem 1. If $\nabla_{\boldsymbol{w}} E_{\boldsymbol{x}|y_{\boldsymbol{x}}=m} \log\left(f_m(\boldsymbol{x}, \boldsymbol{w})\right)$ is $\lambda_{\boldsymbol{x}|y_{\boldsymbol{x}}=m}$-Lipschitz for each $m \in M$. Assume a_n^t satisfies $P(m) = \sum_{n=1}^{N} \frac{D_n a_n^t P_n(m)}{\sum_{n=1}^{N} D_n a_n^t} + \epsilon_t$ for a constant ϵ_t and $\epsilon_t \leq \epsilon\ \forall t$. Then, we have the following inequality for the weight divergence after the t-th aggregation.

$$\begin{aligned} \left\|\boldsymbol{w}_{glb}^{t} - \boldsymbol{w}_{n\_loc}^{t}\right\| \leq q_n^t * [\sum_{n=1}^{N} a_n^t * \left\|\boldsymbol{w}_{glb}^{t-1} - \boldsymbol{w}_{n\_loc}^{t-1}\right\| \\ + \eta \sum_{n=1}^{N} \sum_{m=1}^{M} \left\|P_n(m) - P(m)\right\| \sum_{j=1}^{t-1} a_n^i * g_{\max}\left(\boldsymbol{w}_{n\_loc}^{t-1-n}\right)] \end{aligned} \tag{13}$$

where $g_{\max}(\boldsymbol{w}) = \max_{i=1}^{C} \left\|\nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \log f_i(\boldsymbol{x}, \boldsymbol{w})\right\|$, $a_n = 1 + \eta \sum_{i=1}^{C} P_n(y = i)\lambda_{\boldsymbol{x}|y=i}$, $q_n^t = \frac{D_n}{\sum_{n=1}^{N} D_n}$ and η is the learning rate. Theorem 1 states that, given certain conditions and assumptions, the local DNN model $\boldsymbol{w}_{n_{l}oc}^{t}$ and global model $\boldsymbol{w}_{glb}^{t}$ are both bounded $\forall t \in T$. This bound implies that the stability of FedRank is maintained, ensuring that the learning process converges.

B.1. Proof of Theorem 1

Based on the definition of $\boldsymbol{w}_{glb}^{t}$ and $\boldsymbol{w}_{n\_loc}^{t}$, we have

$$
\begin{aligned}
&\left\|\boldsymbol{w}_{glb}^{t} - \boldsymbol{w}_{n\_loc}^{t}\right\| \\
&= \left\|\sum_{n=1}^{N} \frac{D_n}{\sum_{n=1}^{N} D_n} \boldsymbol{w}_{glb}^{t} - \boldsymbol{w}_{n\_loc}^{t}\right\| \\
&= \| \sum_{n=1}^{N} \frac{D_n}{\sum_{n=1}^{N} D_n} \Bigg(\boldsymbol{w}_{glb}^{t} - \eta \sum_{i=1}^{C} p^{(k)}(y = i) \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{glb}^{t-1}\right)\right] \\
&- \boldsymbol{w}_{n\_loc}^{t-1} + \eta \sum_{i=1}^{C} p(y = i) \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{n\_loc}^{t-1}\right)\right] \Bigg) \\
&\overset{1}{\leq} \left\|\sum_{n=1}^{N} \frac{D_n}{\sum_{n=1}^{N} D_n} \boldsymbol{w}_{glb}^{t-1} - \boldsymbol{w}_{n\_loc}^{t-1}\right\| \\
&+ \eta \left\|\sum_{n=1}^{N} \frac{D_n}{\sum_{n=1}^{N} D_n} \sum_{i=1}^{C} p^{(k)}(y = i) \left(\nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{glb}^{t-1}\right)\right] - \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{n\_loc}^{t-1}\right)\right]\right)\right\| \\
&\overset{2}{\leq} \sum_{n=1}^{N} \frac{D_n}{\sum_{n=1}^{N} D_n} \left(1 + \eta \sum_{i=1}^{C} p^{(k)}(y = i)\lambda_{\boldsymbol{x}|y=i}\right) \left\|\boldsymbol{w}_{glb}^{t-1} - \boldsymbol{w}_{n\_loc}^{t-1}\right\|.
\end{aligned}
$$

Here, inequality 1 holds because for each class $i \in [C], p(y = i) = \sum_{k=1}^{K} \frac{D_n}{\sum_{n=1}^{N} D_n} p^{(k)}(y = i)$, i.e., the data distribution over all the clients is the same as the distribution over the whole population. Inequality 2 holds because we assume $\nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i(\boldsymbol{x}, \boldsymbol{w})\right]$ is $\lambda_{\boldsymbol{x}|y=i}$-Lipschitz. In terms of $\left\|\boldsymbol{w}_{glb}^{t-1} - \boldsymbol{w}_{n\_loc}^{t-1}\right\|$ for client $n \in [N]$, we have

$$
\begin{aligned}
&\left\|\boldsymbol{w}_{glb}^{t-1} - \boldsymbol{w}_{n\_loc}^{t-1}\right\| \\
&= \|\boldsymbol{w}_{glb}^{t-2} - \eta \sum_{i=1}^{C} p^{(k)}(y = i) \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{glb}^{t-2}\right)\right] - \boldsymbol{w}_{n\_loc}^{t-2} + \eta \sum_{i=1}^{C} p(y = i) \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{glb}^{t-2}\right)\right] \| \\
&\leq \left\|\boldsymbol{w}_{glb}^{t-2} - \boldsymbol{w}_{n\_loc}^{t-2}\right\| + \eta \| \sum_{i=1}^{C} p^{(k)}(y = i) \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{glb}^{t-2}\right)\right] \\
&- \sum_{i=1}^{C} p(y = i) \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{n\_loc}^{t-2}\right)\right] \| \\
&\overset{3}{\leq} \left\|\boldsymbol{w}_{glb}^{t-2} - \boldsymbol{w}_{n\_loc}^{t-2}\right\| + \eta \left\|\sum_{i=1}^{C} p^{(k)}(y = i) \left(\nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{glb}^{t-2}\right)\right] - \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{n\_loc}^{t-2}\right)\right]\right)\right\| \\
&+ \eta \| \sum_{i=1}^{C} \left(p^{(k)}(y = i) - p(y = i)\right) \nabla_{\boldsymbol{w}} \mathbf{E}_{\boldsymbol{x}|y=i} \left[\log f_i \left(\boldsymbol{x}, \boldsymbol{w}_{n\_loc}^{t-2}\right)\right] \| \\
&\overset{4}{\leq} \left(1 + \eta \sum_{i=1}^{C} p^{(k)}(y = i) L_{\boldsymbol{x}|y=i}\right) \left\|\boldsymbol{w}_{glb}^{t-2} - \boldsymbol{w}_{n\_loc}^{t-2}\right\| + \eta g_{\max}\left(\boldsymbol{w}_{n\_loc}^{t-2}\right) \sum_{i=1}^{C} \left\|p^{(k)}(y = i) - p(y = i)\right\|.
\end{aligned}
$$

Here, inequality 3 holds because

$$
\begin{aligned}
&\| \sum_{i=1}^{C} p^{(k)}(y=i)\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{glb}^{t-2}\right)\right] - \sum_{i=1}^{C} p(y=i)\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{n\_loc}^{t-2}\right)\right] \| \\
&= || \sum_{i=1}^{C} p^{(k)}(y=i)\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{glb}^{t-2}\right)\right] - \sum_{i=1}^{C} p^{(k)}(y=i)\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{n\_loc}^{t-2}\right)\right] + \\
&\sum_{i=1}^{C} p^{(k)}(y=i)\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{n\_loc}^{t-2}\right)\right] - \sum_{i=1}^{C} p(y=i)\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{n\_loc}^{t-2}\right)\right] || \\
&\le || \sum_{i=1}^{C} p^{(k)}(y=i)\left(\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{glb}^{t-2}\right)\right] - \nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{n\_loc}^{t-2}\right)\right]\right) \| \\
&+ \left\| \sum_{i=1}^{C}\left(p^{(k)}(y=i) - p(y=i)\right)\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\left[\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{n\_loc}^{t-2}\right)\right] \right\|
\end{aligned}
$$

Inequality 4 holds because $g_{\max}\left(\boldsymbol{w}_{glb}^{t-2}\right) = \max_{i=1}^{C}\left\|\nabla_{\boldsymbol{w}}\mathbf{E}_{\boldsymbol{x}|y=i}\log f_i\left(\boldsymbol{x},\boldsymbol{w}_{glb}^{t-2}\right)\right\|$. Based on Eq. (3), let $a_n = 1 + \eta\sum_{i=1}^{C} p^{(k)}(y=i)\lambda_{\boldsymbol{x}|y=i}$, by induction, we have

$$
\begin{aligned}
&\left\|\boldsymbol{w}_{glb}^{t-1} - \boldsymbol{w}_{mT-1}^{(c)}\right\| \\
\le & a_n\left\|\boldsymbol{w}_{glb}^{t-2} - \boldsymbol{w}_{n\_loc}^{t-2}\right\| + \eta g_{max}\left(\boldsymbol{w}_{glb}^{t-2}\right)\sum_{i=1}^{C}\left\|p^{(k)}(y=i) - p(y=i)\right\| \\
\le & \left(a_n\right)^2\left\|\boldsymbol{w}_{glb}^{t-3} - \boldsymbol{w}_{n\_loc}^{t-3}\right\| + \eta\sum_{i=1}^{C}\left\|p^{(k)}(y=i) - p(y=i)\right\|\left(g_{\max}\left(\boldsymbol{w}_{glb}^{t-2}\right) + a_n g_{\max}\left(\boldsymbol{w}_{glb}^{t-2}\right)\right) \\
\le & \left(a_n\right)^{t-1}\left\|\boldsymbol{w}_{glb}^{(m-1)t} - \boldsymbol{w}_{n\_loc}^{(m-1)t}\right\| + \eta\sum_{i=1}^{C}\left\|p^{(k)}(y=i) - p(y=i)\right\|\left(\sum_{j=0}^{t-2}\left(a_n\right)^j g_{\max}\left(\boldsymbol{w}_{glb}^{t-2-j}\right)\right) \\
= & \left(a_n\right)^{t-1}\left\|\boldsymbol{w}_{glb}^{(m-1)t} - \boldsymbol{w}_{n\_loc}^{(m-1)t}\right\| + \eta\sum_{i=1}^{C}\left\|p^{(k)}(y=i) - p(y=i)\right\|\left(\sum_{j=0}^{t-2}\left(a_n\right)^j g_{\max}\left(\boldsymbol{w}_{glb}^{t-2-j}\right)\right)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&\left\|\boldsymbol{w}_{glb}^{t} - \boldsymbol{w}_{n\_loc}^{t}\right\| \le q_n^t * [\sum_{n=1}^{N} a_n^t * \left\|\boldsymbol{w}_{glb}^{t-1} - \boldsymbol{w}_{n\_loc}^{t-1}\right\| \\
&+ \eta\sum_{n=1}^{N}\sum_{m=1}^{M}\|P_n(m) - P(m)\|\sum_{j=1}^{t-1} a_n^i * g_{\max}\left(\boldsymbol{w}_{n\_loc}^{t-1-n}\right)]
\end{aligned}
$$

Hence proved.