FedPLT: Scalable, Resource-Efficient, and Heterogeneity-Aware Federated Learning via Partial Layer Training*

Ahmad Dabaja
LIA/CERI
Avignon University, France
Ahmad.Dabaja@univ-avignon.fr

Rachid El-Azouzi
LIA/CERI
Avignon University, France
Rachid.Elazouzi@univ-avignon.fr

***Abstract*—Federated Learning (FL) has gained significant attention in distributed machine learning by enabling collaborative model training across decentralized devices while preserving data privacy. Although extensive research has addressed statistical data heterogeneity, FL still faces several challenges, including high communication and computation overheads, energy inefficiency, and severe device heterogeneity, which require further investigation. Prior work has addressed these issues through submodel training and partial parameter updates. However, such methods often suffer from inconsistent parameter distributions across clients, inaccurate global loss estimation, and increased bias and variance. In this paper, we propose FedPLT (Federated Learning with Partial Layer Training), a novel and structured partial training approach that divides neural network layers into equal-sized sub-layers and assigns them to clients based on their communication and computational capacities using a fixed assignment strategy. This ensures balanced parameter distribution, reduces variance, and supports varying model sizes across devices, making FedPLT well-suited for resource-constrained environments. In addition, we examine the performance of FedPLT when combined with optimal client sampling and show that this integration enhances federated learning performance by reducing sampling variance under the same communication constraints. Through extensive experiments, we show that FedPLT achieves performance comparable to, or even superior to, that of full-model training (i.e., FedAvg), while requiring significantly fewer trainable parameters per client. It also outperforms existing methods in the literature under highly heterogeneous systems by efficiently adapting to clients' resource constraints and reducing the number of stragglers. Moreover, FedPLT can reduce the trained parameters by up to 65–85%, leading to substantial savings in communication costs, while maintaining the same level of performance as full-model training.**

***Index Terms*—Federated Learning, Partial Training, Communication Efficiency, Energy Efficiency, System Heterogeneity, Straggler Mitigation**

I. Introduction

With the growing necessity of making machine learning more private and secure, especially with the implementation of privacy and security regulations such as GDPR (General Data Protection Regulation) [1] and CCPA (California Consumer Privacy Act) [2], federated learning has emerged as a promising approach to address these concerns.

Federated learning [3] is a decentralized machine learning approach that allows models to be trained on local edge devices, each with its private data, without the need to share the data itself. Instead, local devices share the training parameters of their model with a central server, where they are aggregated to form a single global model. This process is repeated over several rounds, allowing for gradual model improvement and generalization. However, despite its advantages, this approach still faces several significant challenges.

One major challenge is the presence of heterogeneity in local data across clients, leading to high variance of the trained model known as client drift error [4], [5]. This issue arises because devices have different datasets that vary in terms of their source, size, and distribution, causing the global model to struggle to generalize across these diverse datasets [6].

Another challenge for federated learning is system heterogeneity, caused by variability in hardware (CPU, memory) and network connectivity [7]. Edge devices can range from smartphones and computers to IoT sensors, each with its own computational and communication limitations. This problem is further exacerbated by the growing trend of training large models, such as Transformers [8], which are resource-intensive, both in terms of communication and computation, and often exceed the capabilities of many edge devices.

The third challenge lies in the communication process itself. Communication overhead remains a significant bottleneck, as frequent model updates between clients and the central server can saturate bandwidth and increase latency, especially in wireless or mobile networks. During each learning round, multiple clients perform local computations before aggregation, with all these efforts contributing to just a single optimization step, which consumes considerable energy. In practice, several factors influence the overall energy and communication costs, including the model size, the number of participating clients, the volume of locally processed data, and the speed of convergence affected by data heterogeneity across clients.

To address these issues, three main strategies have emerged in the literature: *submodeling*, *partial parameter training*, and *client sampling*. Submodeling methods (e.g., FedDrop [9], HeteroFL [10], and FedRolex [11]) reduce the size of the model trained by each client based on their communication and computational capabilities. Partial parameter training approaches (e.g., FedPMT [12]) maintain the full model structure but update only a subset of parameters during training. Client

*This work is funded and supported by ANR ANR-22-CE23-0024.

sampling strategies, such as Optimal Client Sampling (OCS) [13], aim to reduce overall communication bandwidth usage by selecting a subset of clients to participate in each round.

While these methods help mitigate the impact of device heterogeneity and reduce communication and computing costs, they also introduce new limitations, such as imbalanced parameter distribution across clients, increased variance, unstable convergence, and poor scalability in systems mainly composed of low resource devices.

To address the limitations of existing submodeling and partial parameter training approaches, we propose FedPLT (Federated Learning with Partial Layer Training), a novel framework that provides an effective and structured method for partial training. Unlike previous approaches that introduce noise through random dropout (e.g., FedDrop) or create imbalanced parameter distributions (e.g., HeteroFL, FedRolex, and FedPMT), FedPLT ensures balanced and coordinated training by dividing neural network layers into equal-sized sub-layers and assigning each device a group of sub-layers based on its communication and computational capacities.

The key design of FedPLT is its fixed assignment strategy, which remains constant throughout training. This eliminates variance and instability commonly introduced by dynamic submodel selection methods. Additionally, by ensuring that all sub-layers are uniformly distributed across clients, FedPLT balances parameter updates, leading to a more stable and fair training process. Importantly, FedPLT is flexible, allowing large models to be trained even on low-bandwidth networks and low-resource devices by adapting the proportion of parameters trained per client.

Our main contributions are as follows:

- We introduce FedPLT, a structured partial training scheme that addresses key limitations in existing submodeling and partial training methods in federated learning.
- We conduct extensive experiments to demonstrate that FedPLT achieves improvements over full-model training while reducing the number of trained parameters by 84%, and thus significantly reducing communication costs. Furthermore, FedPLT consistently outperforms other state-of-the-art sub-modeling and partial learning approaches. The performance gap is especially notable on the CIFAR-10 dataset, where competing methods struggle to generalize. More specifically, FedPLT outperforms FedPMT by 4.91%, FedRolex by 6.42%, HeteroFL by 9.26%, and FedDrop by 20.82%.
- We evaluate the effect of combining FedPLT with Optimal Client Sampling (OCS). While OCS reduces communication by selecting fewer clients, integrating it with FedPLT further reduces computation and improves generalization. Under the same communication budget, the combination outperforms OCS alone and reduces variance due to sampling.
- Overall, our findings position FedPLT as a scalable, communication-efficient, and energy-aware solution for real-world federated learning involving large models and highly diverse client capabilities.

II. Related Work

In this section, we review existing literature that addresses communication and computation efficiency, system heterogeneity, and energy-aware training in federated learning (FL) through submodeling, partial training, and client sampling.

Various methods have been proposed to reduce the communication and computational load on client devices, especially under bandwidth and hardware constraints. Broadly, these strategies fall into two main categories:

1) **Sub-modeling Approach:** This approach reduces the communications and computational burden by shrinking the global model with a factor that depends on the client's capabilities. This is achieved by omitting certain training parameters, resulting in smaller sub-models that are less resource demanding. Several works follow this approach, including HeteroFL [10], FedDrop [9], and FedRolex [11]. The difference between these methods lies in the way sub-modeling is applied.
2) **Partial Parameters Training Approach:** Unlike sub-modeling, this method keeps all the model parameters intact but updates only a subset of them during training. This reduces communication and computation without altering the overall model architecture. To the best of our knowledge, FedPMT [12] is the only method that follows this approach at the time of writing.

Each of the two approaches has distinct advantages and limitations. During inference, the full global model is used to evaluate loss and accuracy on client data. However, during training, the sub-modeling approach uses reduced models that omit some parameters. As a result, the forward pass computes loss using an incomplete model, leading to inaccurate gradients and suboptimal global updates. Empirical studies have shown that aggressive shrinking (i.e., high reduction ratios) often degrades performance, making extreme sub-modeling impractical.

In contrast, the partial parameter training approach performs the forward pass using the full model but updates only a subset of parameters during backpropagation. This reduces gradient error while preserving the model structure.

However, this comes with a slightly higher overhead. While sub-modeling applies both forward and backward passes on the reduced model, partial training computes the forward pass on the full model and updates only selected parameters. Additionally, the full model is transmitted to the client, although only updated parameters are sent back. Sub-modeling, by contrast, exchanges the reduced model in both directions.

In particular, each sub-modeling approach offers unique trade-offs. FedDrop [9] randomly drops neurons in fully connected layers at each round, following a Bernoulli distribution. While this reduces communication and computation, the randomness introduces noise and variance across rounds, leading to unstable gradients, fluctuating loss, and slower convergence.

HeteroFL [10] adopts a fixed dropout mask applied before training, where neurons on one side of the layer are dropped while keeping the others, avoiding the randomness of FedDrop and improving stability. However, it results in imbalanced parameter distribution across clients where frequently retained

neurons are updated by all clients, while others are only trained by high-resource and bandwidth devices. This bias can harm performance, particularly when training on complex datasets using small models that struggle to generalize across heterogeneous data distributions.

FedRolex [11] addresses this by gradually shifting the dropout mask, exposing all neurons to diverse data over time. This reduces parameter bias but requires many rounds to complete a full rotation, increasing convergence time and introducing the risk of overwriting earlier updates, which may lead to a form of catastrophic forgetting. Additionally, FedRolex reintroduces variance from FedDrop due to the use of shifting masks.

FedPMT [12] is a partial parameter training approach that generates a limited number of training configurations by freezing the shallow dense layers of a fully connected network. In this method, clients with high resources and good bandwidth train the full model, while low-resource and bandwidth clients update only the deeper layers. However, FedPMT has several limitations:

- Limited Flexibility in Training Proportions: The number of valid partial training configurations is constrained by the number of dense layers. Since shallow layers contain most of the parameters, this results in large gaps between these proportions, forcing clients to either overtrain or under-utilize their capacity, thus promoting stragglers.
- Unbalanced Training Across Clients: FedPMT prioritizes deeper layers, leading to an uneven distribution of trainable parameters. High-resource and bandwidth clients train most of the model, while weaker clients update only the final layers, creating imbalance and introducing training bias.

A common limitation of all these methods is that they become impractical for training large models in systems where all clients have limited communications and computational capacities. These approaches typically assume that at least some clients can train the full model. However, in environments where no device has sufficient capacity, such as in the case of training large-scale models like LLMs, these strategies break down, rendering them unsuitable for fully resource-constrained systems.

Another widely studied strategy is *client sampling*, where only a subset of clients participates in each round. This both reduces the bandwidth and resource usage but can introduce gradient noise and oscillations due to data heterogeneity, potentially slowing convergence.

One recent method is Optimal Client Sampling (OCS) [13], which samples, under a fixed communication budget, clients that contribute with larger gradients and higher data counts. This is done by minimizing the variance in the aggregated model update due to partial participation as follows:

$$\min_{\{p_k^t\}} \mathbb{E}_{A^t} \left[\left\| \sum_{k \in A^t} \frac{n_k}{p_k^t} U_k^t - \sum_{k=1}^{K} n_k U_k^t \right\|^2 \right] \tag{1}$$

subject to the communication constraint κ such that:

$$\sum_{k=1}^{K} p_k^t = \kappa, \tag{2}$$

where n_k is the number of data samples held by client k, U_k^t is the client's gradient, p_k^t is the client's sampling probability, and A^t is the set of sampled clients.

Although OCS shows better performance and reduces communication costs, it requires computing gradients on all clients, making it computationally expensive.

A crucial yet unexplored question, to our knowledge, is which approach, client sampling or submodeling/partial model training, more effectively reduces communication and computational costs in heterogeneous federated learning systems.

III. The Proposed Approach

A. Federated Learning Overview

We consider a standard Federated Learning (FL) setting with a central server and K clients $\{C_k\}_{k=1}^K$, each holding a local dataset D_k of size n_k. The objective is to minimize a global loss over the union of local datasets:

$$F(W) = \frac{1}{\sum_k n_k} \sum_k n_k \cdot F_k(W),$$

where $F_k(W)$ is the empirical loss over client k's data.

The typical optimization algorithm is *FedAvg* (Federated Averaging) [3]. At each round t, clients receive a copy of the global model W^t and perform τ local SGD updates:

$$w_k^{t,s+1} = w_k^{t,s} - \eta_k \nabla F_k(w_k^{t,s}), \quad s = 0, \ldots, \tau - 1.$$

Clients send their local models $w_k^{t,\tau}$ back to the server, which updates the global model via weighted aggregation:

$$W^{t+1} = \frac{1}{\sum_k n_k} \sum_k n_k \cdot w_k^{t,\tau}.$$

This process is repeated for R communication rounds until convergence.

B. Motivation and Design: Partial Layer Training

Partial parameter training is more stable and accurate than sub-modeling. However, existing approaches like FedPMT suffer from unbalanced parameter distribution and limited flexibility in selecting training proportions. To address these limitations, we propose a more structured and balanced approach based on partial layer training.

The core idea of our scheme is to divide each layer into equal-sized sublayers. The server assigns different proportions of these sublayers to clients based on their communication and computational capacities. This assignment is performed once at the beginning of training and ensures uniform coverage of all sublayers across clients.

Our method applies a layer-freezing technique similar to FedPMT but in a more structured and balanced manner. It enables longitudinal sublayer allocation and supports flexible training proportions, allowing clients with different capacities to participate effectively. Figure 1 compares two submodels of a fully connected neural network generated by FedPMT and FedPLT with the same trained parameter ratio.

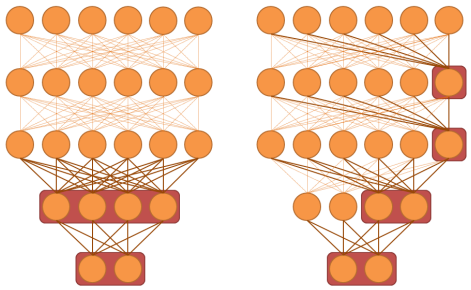

Fig. 1. Comparison of two partial parameter training configurations with the same proportion of trained parameters (brown blocks), generated using FedPMT (left) and FedPLT (right).

C. FedPLT Algorithm and Aggregation

In this section, we present FedPLT using dense layers, which are common across various models and tasks and are communication and computation intensive. However, our method can extends to convolutional and Transformer layers.

We consider a fully connected model with trainable parameters W composed of L dense layers. Before training begins, the server assesses the computational capacity of each client C_k and assigns it a training proportion r_k, representing the fraction of total parameters it can handle. To implement this, the server constructs a layer-wise proportion vector $Q_k = \{q_{k1}, q_{k2}, \ldots, q_{kL}\}$, where q_{kl} denotes the share of layer l assigned to client C_k. The total proportion constraint is defined as: $r_k = \frac{1}{\sum_l H_l} \sum_l q_{kl} \cdot H_l$, where H_l is the number of parameters in layer l. This ensures balanced training across sub-layers while respecting each client's communication and computation limits.

Multiple Q_k vectors can satisfy this constraint. Based on empirical observations, we identify two effective assignment strategies: (1) assigning the full classification layer to all clients and distributing equal proportions across the other layers (e.g., $Q_k = (\mu, \mu, \mu, 1)$), or (2) gradually increasing the assigned proportions toward deeper layers. Our results suggest that the structural pattern of sub-layer assignment significantly impacts performance.

Once Q_k is determined, the server generates sub-layer assignments $\Upsilon_k = \{\upsilon_{k1}, \upsilon_{k2}, \ldots, \upsilon_{kL}\}$ for each client C_k, where υ_{kl} is the set of sub-layers from layer l. Figure 1 illustrates a client with $r_k \approx 30\%$ and $Q_k = \{1/6, 1/6, 1/2, 1\}$. Υ_k is thus shown as brown blocks.

During training, each client updates only its assigned sub-layers and sends them to the server. The server performs aggregation per sub-layer using a weighted average over the clients that trained it, based on their local dataset sizes.

The update for sub-layer $\upsilon_{l,h}$ in layer l at round $t+1$ is computed as:

$$\upsilon_{l,h}^{t+1} = \frac{\sum_{C_k \in \mathcal{S}_{l,h}} n_k \cdot \upsilon_{k,l,h}^{t}}{\sum_{C_k \in \mathcal{S}_{l,h}} n_k} \tag{3}$$

where:

- $\upsilon_{l,h}^{t+1}$: updated global parameters for sub-layer h in layer l.
- $\upsilon_{k,l,h}^{t}$: local parameters for sub-layer h from client C_k at round t.
- $\mathcal{S}_{l,h}$: set of clients that trained sub-layer h in layer l.

The complete FedPLT procedure is detailed in Algorithm 1.

Algorithm 1 FedPLT Scheme

Input: Initial model W^0, number of layers L, clients $C_1, ..., C_K$, rounds R

Client Initialization:
for each client C_k **do**
  Assign training proportion r_k
  Compute proportion vector $Q_k = \{q_{k1}, ..., q_{kL}\}$
  Generate sub-layer assignments $\Upsilon_k = \{\upsilon_{k1}, ..., \upsilon_{kL}\}$
end for

Federated Training:
for each round $t = 1, ..., R$ **do**
  for each client C_k in parallel **do**
    Receive W^t
    Train assigned sub-layers Υ_k; freeze others
    Send updated parameters to server
  end for
  for each sub-layer $\upsilon_{l,h}$ **do**
    Aggregate updates:
    $\upsilon_{l,h}^{t+1} = \frac{\sum_{C_k \in \mathcal{S}_{l,h}} n_k \cdot \upsilon_{k,l,h}^{t}}{\sum_{C_k \in \mathcal{S}_{l,h}} n_k}$
  end for
end for

Output: Final model W^R

The convergence of FedPLT is achieved using a decaying learning rate, with a convergence rate of $\mathcal{O}(\frac{1}{t})$ under standard assumptions of strong convexity and smoothness. The proof of convergence is available in [14].

D. Optimal Client Sampling with Partial Layer Training

In Federated Learning, client sampling plays a key role in balancing communication cost and convergence stability. While prior work on Optimal Client Sampling (OCS) [13] minimizes the variance of the aggregated model between full and partial participation under a fixed communication cost, our setting introduces an additional challenge: clients train different fractions of the model due to partial layer assignment.

To adapt OCS to our FedPLT scheme, we replace the original constraint $\sum_{k=1}^{K} p_k^t = \kappa$ with a FedPLT-aware constraint:

$$\sum_{k=1}^{K} r_k p_k^t = \kappa,$$

where r_k denotes the ratio of the model trained by client k.

Solving the resulting optimization problem using Lagrange multipliers and KKT conditions yields modified selection probabilities:

$$p_k^t = \begin{cases} \frac{\kappa - \sum_{j=1}^{K} r_j + \sum_{j \in \mathcal{O}} r_j}{\sqrt{r_k}} \cdot \frac{n_k \|U_k^t\|}{\sum_{j \in \mathcal{O}} \sqrt{r_j} d_j \|U_j^t\|}, & \text{if } k \in \mathcal{O} \\ 1, & \text{if } k \in K - \mathcal{O} \end{cases} \tag{4}$$

where the set $\mathcal{O}$ is defined by:

$$\mathcal{O} = \left\{ k \in \mathbb{N} \Big/ \sqrt{r_k} n_k \|U_k^t\| < \frac{\sum_{j\in\mathcal{O}} \sqrt{r_j} d_j \|U_j^t\|}{\kappa - \sum_{j=1}^{K} r_j + \sum_{j\in\mathcal{O}} r_j} \right\}. \quad (5)$$

This formulation ensures that client selection remains variance-optimal while accounting for each client's communication cost under FedPLT.

IV. Experimentation and Results

We evaluate FedPLT through three experiments that analyze its strength and performance:

- **Demonstrate the Strength of FedPLT:** We evaluate FedPLT on a resource constraint system of homogeneous devices where clients train only a small proportion of the model parameters (r_k). The performance is compared to the FedAvg as a baseline.
- **Benchmarking FedPLT in a Heterogeneous Setting:** We assess FedPLT against existing submodeling and partial training methods in a realistic scenario with highly heterogeneous clients to evaluate its performance in a more realistic federated learning scenario.
- **Experiment 3: FedPLT and Optimal Sampling:** We study the effect of incorporating FedPLT into optimal sampling, exploring potential performance gains and trade-offs by comparing optimal sampling with and without FedPLT.

In all experiments, we use a fully connected neural network with four layers (Input $\rightarrow$ 512 $\rightarrow$ 256 $\rightarrow$ 128 $\rightarrow$ Output) to classify the Fashion MNIST [15] and CIFAR-10 [16] datasets. Data is distributed across 50 clients (100 in Experiment 3) using a Dirichlet distribution with $\alpha = 0.2$ ($\alpha = 0.1$ for Experiment 3). In Experiments 1 and 2, we use mini-batch SGD with 1 local epoch per round and a batch size of 64. In Experiment 3, we apply full-batch gradient descent with 3 local epochs. Training is conducted for 300, 500, or 1000 rounds depending on the dataset and experiment. The learning rate is fixed at 0.01. To mitigate randomness, each experiment is repeated 10 times with different random seeds, and results are averaged.

A. Demonstrating the Strength of FedPLT

In this experiment, we train 50 homogeneous clients, each with a fixed training proportion of $r_k = 29\%$ for Fashion MNIST. Three different layer proportion vectors are tested to generalize the results: $Q_k = (20, 45, 76, 80)$, $(10, 73, 86, 90)$, and $(25, 29, 78, 100)$. For CIFAR-10, we used the same layer proportion vectors, which correspond to the proportions 23%, 16%, and 26%. We compare these to a full-model FedAvg baseline ($Q_k = (100, 100, 100, 100)$).

Figure 2 shows that training only a portion of the model on each client achieves comparable, and sometimes better, generalization than full-model training. This suggests that FedPLT may introduce a regularization effect, improving performance while significantly reducing communication and computation costs. For instance, on Fashion MNIST, training with $Q_k = (20, 45, 76, 80)$ led to a 3.32% increase in accuracy compared to full-model training. On CIFAR-10, the more challenging dataset, the maximum drop in accuracy among the three configurations was only 1.44%, but with achieving up to a 74% reduction in the parameter trained and thus in the resource utilizated per client.

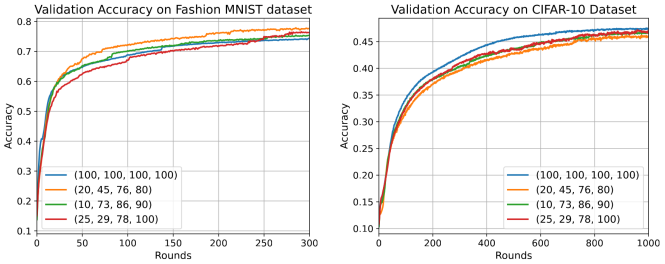

Fig. 2. The accuracy curves obtained from running FedPLT on homogeneous and low resources system with four different Q_k vectors and for both Fashion MNIST and CIFAR-10 datasets, along with that of the fully trained model.

These results also highlight FedPLT's strong performance in resource-constrained environments and its potential for training large models (e.g., LLM) on low-power systems such as IoT and mobile devices.

B. Benchmarking FedPLT against Existing Methods in a Heterogeneous Setting

To evaluate the performance of FedPLT in a realistic heterogeneous FL environment, we simulate device diversity by assigning 10% of the clients with full capacity ($r_k = 100\%$), 30% with moderate capacity ($r_k = 29\%$ - for Fashion MNIST), and 60% with low capacity ($r_k = 6\%$). We have also simulated each of the following schemes, FedPMT, HeteroFL, FedDrop, FedRolex, and FedAvg, with equivalent configuration.

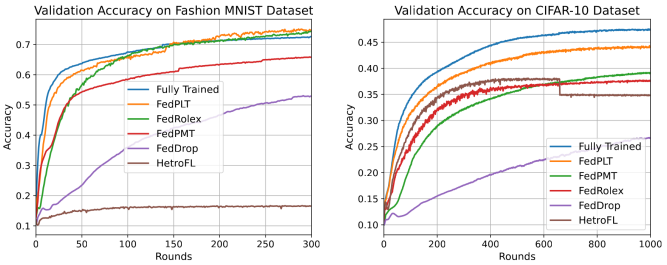

Fig. 3. Validation accuracy curves comparing FedPLT with existing methods in a heterogeneous and low-resource FL system.

Figure 3 highlights several key insights. First, FedPLT achieves validation accuracy comparable to FedAvg on CIFAR-10 (44.07% vs. 47.42%) and even exceeds it on Fashion MNIST by 2.17%. This confirms the findings from the homogeneous setting (Experiment 1) and demonstrates that they extend to highly heterogeneous and resource-constrained systems.

Second, FedPLT consistently outperforms other state-of-the-art submodeling and partial training schemes across both datasets. The performance gap is especially pronounced on the more challenging CIFAR-10 dataset, where competing

methods struggle to generalize. Notably, FedPLT also maintains a convergence rate similar to full-model training. After 1000 rounds on CIFAR-10, FedPLT outperformed FedPMT by 4.91%, FedRolex by 6.42%, HeteroFL by 9.26%, and FedDrop by 20.82%.

These results, along with FedPLT's theoretical advantage in mitigating stragglers more effectively than existing schemes, position it as a superior approach. By better aligning model complexity with device capabilities, FedPLT enables more efficient and performant training across the system.

Overall, FedPLT presents a scalable, resource-efficient, and heterogeneity-aware FL solution that outperforms state-of-the-art methods in both accuracy and system-wide efficiency.

C. FedPLT and Optimal Client Sampling

In this experiment, we use both the original Optimal Client Sampling (OCS) formulation, where κ denotes the number of selected clients and ignores r_k, and our FedPLT-aware formulation, where κ corresponds to the actual communication cost. We evaluate both on two heterogeneous systems: one with 10% of clients training the full model and 90% training 20% of the model, and another with 90% of clients training 50% of the model. In all settings, we set $\kappa = 5$.

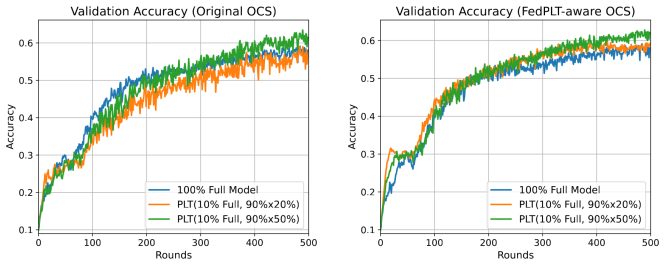

Fig. 4. Validation accuracy curves comparing OCS with and without FedPLT. Left: original OCS formulation. Right: FedPLT-aware OCS.

With the original OCS formulation, combining FedPLT with OCS achieves higher accuracy in the high-resource configuration, where 10% of clients train the full model and 90% train 50% of it, improving accuracy by +3.51% over full-model training. In contrast, the low-resource configuration, where 90% train 20% of the model, performs slightly worse, with a 1.11% drop. Notably, these results are obtained with lower communication costs: 36.25% less in the high-resource setting and 43% less in the low-resource setting, compared to full-model OCS.

Under the FedPLT-aware OCS formulation, where all setups use the same communication budget, FedPLT outperforms full-model training by +3.84% (50%) and +1.29% (20%). This confirms that incorporating FedPLT improves generalization even under an equal communication budget.

Overall, combining OCS with FedPLT improves generalization and delivers better accuracy with lower resource usage. Across all three experiments, FedPLT proves to be effective both with and without client sampling, confirming its flexibility under diverse system constraints.

V. Conclusion

This paper proposed FedPLT, a structured partial layer training scheme for federated learning that addresses system heterogeneity and communication and computation bottlenecks. Through extensive experiments, we demonstrated that FedPLT offers strong generalization while significantly reducing resource usage. It consistently outperforms existing sub-modeling and partial training methods, mitigates stragglers' impact, and integrates effectively with client sampling, making it scalable and flexible for real-world FL deployments.

References

[1] E. Union, "General data protection regulation (gdpr)," *Official Journal of the European Union*, L119, pp. 1-88, Apr. 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj. [Accessed: Sep. 9, 2024], 2016.

[2] S. of California, "California consumer privacy act (ccpa)," *California Legislative Information*, AB-375, Jun. 2018. [Online]. Available: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375. [Accessed: Sep. 9, 2024], 2018.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 54, pp. 1273–1282, 2017.

[4] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJxNAnVtDS

[5] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5132–5143. [Online]. Available: https://proceedings.mlr.press/v119/karimireddy20a.html

[6] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, "Heterogeneous federated learning: State-of-the-art and research challenges," *ACM Comput. Surv.*, vol. 56, no. 3, oct 2023. [Online]. Available: https://doi.org/10.1145/3625558

[7] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv: Learning*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:59316566

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[9] D. Wen, K.-J. Jeon, and K. Huang, "Federated dropout—a simple approach for enabling federated learning on resource constrained devices," *IEEE wireless communications letters*, vol. 11, no. 5, pp. 923–927, 2022.

[10] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://arxiv.org/abs/2010.01264

[11] S. Alam, L. Liu, M. Yan, and M. Zhang, "Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction," *Advances in neural information processing systems*, vol. 35, pp. 29 677–29 690, 2022.

[12] H. Wu, P. Wang, and C. V. A. Narayana, "Straggler-resilient federated learning: Tackling computation heterogeneity with layer-wise partial model training in mobile edge network," *arXiv preprint arXiv:2311.10002*, 2023. [Online]. Available: https://arxiv.org/abs/2311.10002

[13] H. Zhang, Z. Li, Z. Gong, M. Siew, C. Joe-Wong, and R. El-Azouzi, "Poster: Optimal variance-reduced client sampling for multiple models federated learning," in *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*, 2024, pp. 1446–1447.

[14] "Fedplt github respiratory," https://github.com/AhmadDabaja/FedPLT.

[15] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[16] A. Krizhevsky and G. Hinton, "Cifar-10 and cifar-100 datasets," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html