Communication-Efficient and Interoperable Distributed Learning

Mounssif Krouka and Mehdi Bennis, Fellow, IEEE

Abstract

Collaborative learning across heterogeneous model architectures presents significant challenges in ensuring interoperability and preserving privacy. We propose a communication-efficient distributed learning framework that supports model heterogeneity and enables modular composition during inference. To facilitate interoperability, all clients adopt a common fusion-layer output dimension, which permits each model to be partitioned into a personalized base block and a generalized modular block. Clients share their fusion-layer outputs, keeping model parameters and architectures private. Experimental results demonstrate that the framework achieves superior communication efficiency compared to federated learning (FL) and federated split learning (FSL) baselines, while ensuring stable training performance across heterogeneous architectures.

Index Terms

Distributed learning, communication efficiency, cross-vendor deployment, interoperability.

I. Introduction

Driven by advances in wireless technologies, the number of interconnected devices has grown rapidly, leading to the generation and transmission of massive volumes of data. This surge has enabled a wide range of data-driven artificial intelligence (AI) applications [1]–[3]. In centralized AI systems, where raw data is transmitted to remote servers for training, communication overhead is high, and data privacy is compromised. To address these challenges, Federated Learning (FL) was introduced, allowing clients to keep their local data and share only model updates with a central server [4]–[6]. However, transmitting the full model updates in each training round leads to high communication costs. Additionally, FL requires all clients to adopt a common

M. Krouka and M. Bennis are with the Centre for Wireless Communications (CWC), University of Oulu, Finland (emails: {mounssif.krouka, mehdi.bennis}@oulu.fi). Preprint version. Submitted for peer review.

model architecture, limiting flexibility and promoting vendor lock-in, which impedes support for heterogeneous models. To reduce communication overhead, Split Learning (SL) serves as a more efficient alternative. In SL, the client processes a portion of the model and transmits the intermediate output from a designated cut-layer to the server, which completes the remaining computation [7]–[10]. As a result, model parameters are not exchanged.

When extended to multiple clients, Federated Split Learning (FSL) enables collaborative training with distributed client-side model partitions and a shared server-side model [11]–[13]. However, FSL requires clients to rely on the server's model part during both training and inference. This exposes server-side model parameters and architecture to the server, constraining modular AI designs and limiting component-level competition [14].

In this work, we propose a communication-efficient distributed learning architecture that supports heterogeneous AI models across clients, while enabling cross-vendor interoperability and model composition during inference. The clients agree on a common output dimension of a fusion-layer, which partitions each model into a lower-level base block and an upper-level modular block. This architecture enables end-to-end training without exposing model parameters, gradients, or architecture details to other clients or the server. Furthermore, modular blocks can be flexibly interchanged across clients via the fusion layer, facilitating model composition during inference. We highlight the main contributions as follows:

- We propose a novel two-stage training algorithm that enables personalization of the base block using local data and generalization of the modular block using concatenated fusion-layer outputs from the clients.
- We allow multiple local updates to the base block before transmitting fusion-layer outputs, unlike FSL which performs only a single update per communication round, thereby reducing communication frequency and overhead.
- We enable heterogeneous and proprietary model designs by ensuring that the model parameters, gradients, and architecture of each client remain private and restricted to the client side, with only the output of the fusion-layer shared with the server.
- We leverage the same fusion-layer output dimension among the clients to standardize the interface between model partitions, enabling modular block interoperability across heterogeneous client models and supporting flexible multi-vendor deployments.
- We empirically demonstrate that our framework significantly reduces communication overhead compared to FL and FSL, while maintaining high accuracy. We further validate its

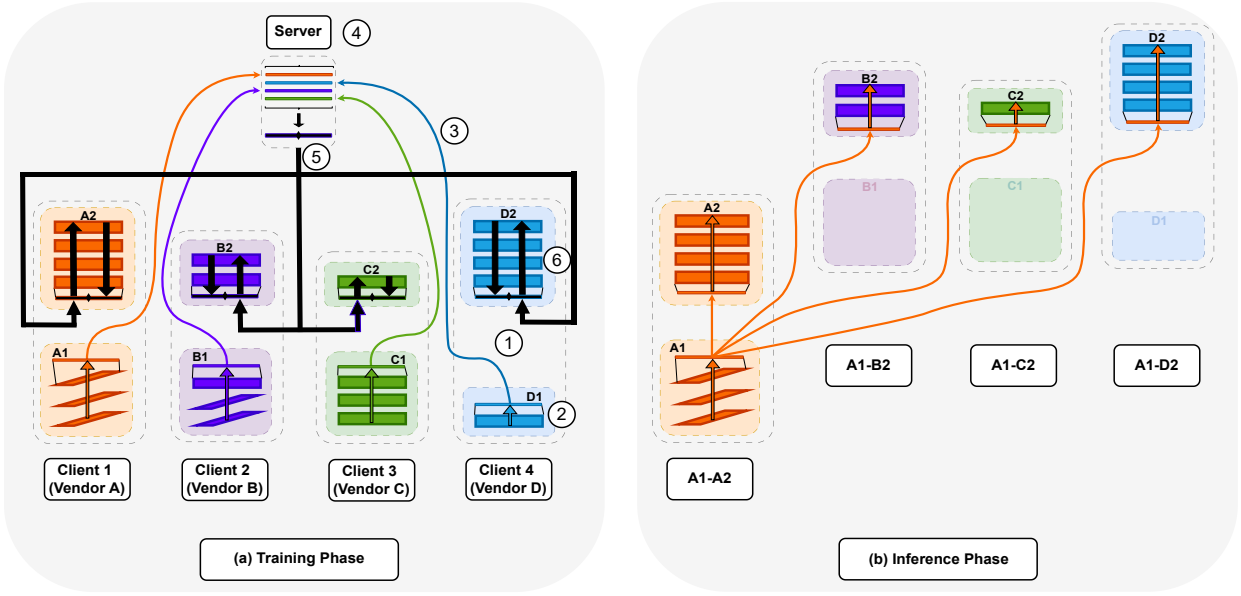

Fig. 1. Overview of the proposed framework's training and inference phases. (a) Training: local base block updates and server-side concatenation of fusion-layer outputs for modular block training. (b) Inference: base block combined with modular blocks from different clients to enable cross-client composition.

robustness to model heterogeneity and its support for seamless modular block interoperability during inference across diverse client models.

II. SYSTEM MODEL

We consider a distributed learning system coordinated by a remote server with N clients, each having a local dataset and a model with a client-specific architecture. To enable collaboration across clients while preserving model heterogeneity and ownership, we adopt an architecture that partitions each client's model into two components: a base block and a modular block, separated by a fusion-layer with a common output dimension across all the clients. The base block layers are from the input until the fusion-layer, and the modular block layers constitute the remaining part until the output layer.

A. Training Phase

The end-to-end training process is shown in Fig. 1a, where we assume that each client deploys a model architecture from a different vendor. We denote the base block and the modular block

of a given vendor (e.g., vendor X) as $X1$ and $X2$, respectively. We follow the numbering in Fig. 1a and describe the training steps in this way:

- Step 1: Each client performs τ update steps to the base block parameters.
- Step 2: Clients start forward propagation on the base block until the fusion-layer.
- Step 3: Each client sends its fusion-layer output and labels to the server.
- Step 4: On the server side, the server concatenates the outputs received from the clients.
- Step 5: The server broadcasts the result to the clients.
- Step 6: Each client uses the data received from the server as input to the modular block and updates its parameters.

B. Inference Phase

After training is complete, each client retains a personalized base block trained on local data and a generalized modular block trained using the concatenated outputs of the fusion layers of all clients. A common fusion-layer output dimension is required across clients to ensure compatibility and support modular composition. This standardization enables vendors to design model architectures independently, without concern for cross-vendor compatibility. Fig. 1b illustrates an example of modular composition, where the base block of vendor A, referred to as $A1$, is deployed with the modular blocks $A2$, $B2$, $C2$, and $D2$, which correspond to the vendors A, B, C, and D, respectively.

Our proposed solution aims to support heterogeneous model architectures and to empower vendors to retain ownership of their model architectures, while enabling the composition and interoperability of modular blocks from different vendors.

III. Problem Formulation and Proposed Scheme

Building on the proposed architecture introduced in Section II, we now formalize the challenges in existing distributed learning paradigms and introduce our proposed solution.

A. Problem Formulation

Formally, all clients collaboratively optimize a global model $\boldsymbol{\theta}$ by minimizing the weighted sum of their local empirical risks:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \; F(\boldsymbol{\theta}) = \sum_{k=1}^{N} \frac{d_k}{d} F_k(\boldsymbol{\theta}), \tag{1}$$

where $d = \sum_{k=1}^{N} d_k$ is the total number of samples across all clients and d_k is the number of samples at client k. The local objective of client k is given by

$$F_k(\boldsymbol{\theta}) = \mathbb{E}_{(x_k, y_k) \sim \mathcal{D}_k}[\ell(f(x_k; \boldsymbol{\theta}), y_k)], \tag{2}$$

where (x_k, y_k) denotes the input-label pairs drawn from its dataset $\mathcal{D}_k$, ℓ is the loss function, and $\boldsymbol{\Theta}$ is the feasible set of model parameters.

During the communication round t, the server sends the global model $\boldsymbol{\theta}^t$ to each client k, which then performs τ local Stochastic Gradient Descent (SGD) steps as follows [15]:

$$\boldsymbol{\theta}_k^{t,s} = \boldsymbol{\theta}_k^{t,s-1} - \eta \nabla F_k(\boldsymbol{\theta}_k^{t,s-1}), \quad \forall s \in \{1, 2, \cdots, \tau\}, \tag{3}$$

with $\boldsymbol{\theta}_k^{t,0} = \boldsymbol{\theta}^t$. Then, the clients send their local models to the server for aggregation:

$$\boldsymbol{\theta}^{t+1} = \sum_{k=1}^{N} \frac{d_k}{d} \boldsymbol{\theta}_k^{t,\tau}. \tag{4}$$

However, transmitting the full model parameters $\boldsymbol{\theta}_k^{t,\tau}$ to the server incurs high communication overhead, in addition to composing a global model $\boldsymbol{\theta}^{t+1}$ that must be shared among clients, undermining customization and ownership.

To reduce per-round upload sizes, FSL enables partitioning the model at the cut-layer into a client-side model $f_c(x_k; \boldsymbol{\theta}_{c,k})$, with client-specific parameters $\boldsymbol{\theta}_{c,k}$, and a server-side model $f_s(h_k; \boldsymbol{\theta}_s)$, with shared server parameters $\boldsymbol{\theta}_s$, where

$$h_k = f_c(x_k; \boldsymbol{\theta}_{c,k}) \tag{5}$$

denotes the output of the cut-layer that is sent from client k to the server. The server then continues the forward propagation to generate the predicted labels as follows:

$$\tilde{y}_k = f_s(h_k; \boldsymbol{\theta}_s). \tag{6}$$

Transmitting the output of the cut-layer yields a smaller communication cost per round. Nonetheless, FSL allows the server to retain part of the model, with access to both the architecture and the parameters. Moreover, local end-to-end inference is not supported, as the server-side model is required for all clients, which hinders interoperability between clients.

B. Proposed Scheme

We introduce a novel interoperable Federated Learning (IFL) approach that enables clients to retain their full model architecture and parameters locally while collaboratively training across heterogeneous models. Each client k decomposes its model into a base block $f_{b,k}(\cdot;\boldsymbol{\theta}_{b,k})$ and a modular block $f_{m,k}(\cdot;\boldsymbol{\theta}_{m,k})$, where $\boldsymbol{\theta}_{b,k}$ and $\boldsymbol{\theta}_{m,k}$ denote the parameters of the respective blocks. These blocks are connected by a fusion-layer whose output is denoted as z_k. Crucially, only the fusion-layer output z_k (intermediate representation) and the corresponding labels y_k are shared with the server.

Our training process alternates between updating the base blocks using local data samples and updating modular blocks using concatenated fusion-layer outputs from all clients. This approach decouples personalization (base block) from generalization (modular block), while preserving model ownership and reducing communication costs. In what follows, we describe the update details of both blocks during communication round $t \in \{1, 2, \ldots, T\}$. The detailed pseudocode can be found in Algorithm 1.

1) Base Block Update: Each client k performs forward and backward propagation over a sequence of τ mini-batches drawn from its local dataset $\mathcal{D}_k$. During this local update phase, only the parameters of the base block are optimized via SGD, while the modular block remains fixed. The base block parameters $\boldsymbol{\theta}_{b,k}$ are updated according to

$$\boldsymbol{\theta}_{\mathrm{b},k}^{t+1} = \boldsymbol{\theta}_{\mathrm{b},k}^{t} - \eta_{\mathrm{b}} \sum_{s=1}^{\tau} \Delta_{\boldsymbol{\theta}_{\mathrm{b},k}} \ell(f_{\mathrm{m},k}(f_{\mathrm{b},k}(x_k^s; \boldsymbol{\theta}_{\mathrm{b},k}^t); \boldsymbol{\theta}_{\mathrm{m},k}^t), y_k^s), \tag{7}$$

where η_{b} is the learning rate, $\Delta_{\boldsymbol{\theta}_{\mathrm{b},k}} \ell(\cdot)$ denotes the gradient of the loss with respect to the base block parameters, and (x_k^s, y_k^s) represents the s^{th} mini-batch sampled from $\mathcal{D}_k$.

Algorithm 1 Interoperable Federated Learning (IFL)

Initialize: $\boldsymbol{\theta}^0_{\mathrm{b},k}$, $\boldsymbol{\theta}^0_{\mathrm{m},k}$ for each client $k = 1$ to N
for each communication round $t = 0$ to $T-1$ **do**
  Base Block Update
  for each client $k = 1$ to N **in parallel do**
    for each local step $s = 1$ to τ **do**
      Sample mini-batch (x_k, y_k) from local dataset $\mathcal{D}_k$
      $\hat{y}_k \leftarrow f_{\mathrm{m},k}(f_{\mathrm{b},k}(x_k; \boldsymbol{\theta}^t_{\mathrm{b},k}); \boldsymbol{\theta}^t_{\mathrm{m},k})$
      $\ell_k \leftarrow \mathrm{Loss}(\hat{y}_k, y_k)$
      $\boldsymbol{\theta}^t_{\mathrm{b},k} \leftarrow \boldsymbol{\theta}^t_{\mathrm{b},k} - \eta_b \nabla_{\boldsymbol{\theta}_{\mathrm{b},k}} \ell_k$
    end for
    $\boldsymbol{\theta}^{t+1}_{\mathrm{b},k} \leftarrow \boldsymbol{\theta}^t_{\mathrm{b},k}$
  end for
  Fusion-Layer Output Transmission
  for each client $k = 1$ to N **in parallel do**
    Sample fresh mini-batch (x_k, y_k)
    $z_k \leftarrow f_{\mathrm{b},k}(x_k; \boldsymbol{\theta}^{t+1}_{\mathrm{b},k})$
    Send (z_k, y_k) to server
  end for
  Server Concatenation and Broadcast
  Server collects $Z = \{z_1, \ldots, z_N\}$ and $Y = \{y_1, \ldots, y_N\}$
  Server broadcasts (Z, Y) to all clients
  Modular Block Update
  for each client $k = 1$ to N **in parallel do**
    for each $i = 1$ to N **do**
      $\hat{y}_i \leftarrow f_{\mathrm{m},k}(z_i; \boldsymbol{\theta}^t_{\mathrm{m},k})$
      $\ell_i \leftarrow \mathrm{Loss}(\hat{y}_i, y_i)$
      $\boldsymbol{\theta}^t_{\mathrm{m},k} \leftarrow \boldsymbol{\theta}^t_{\mathrm{m},k} - \eta_m \nabla_{\boldsymbol{\theta}_{\mathrm{m},k}} \ell_i$
    end for
    $\boldsymbol{\theta}^{t+1}_{\mathrm{m},k} \leftarrow \boldsymbol{\theta}^t_{\mathrm{m},k}$
  end for
end for

TABLE I

COMPARISON OF KEY FEATURES ACROSS FL, FSL, AND THE PROPOSED IFL FRAMEWORK

Feature	**FL**	**FSL**	**IFL**
Client parameters remain private	✗	✓	✓
Local end-to-end inference	✓	✗	✓
Lightweight uplink update	✗	✓	✓
Multiple updates per round	✓	✗	✓
Full model architecture privacy	✗	✗	✓
Heterogeneous model support	✗	✗	✓
Cross-client model composition	✗	✗	✓

After the update of the base block parameters, each client k samples a fresh mini-batch (x_k, y_k) of size B and performs forward propagation on the base block as follows

$$z_k = f_{\mathrm{b},k}(x_k; \boldsymbol{\theta}_{\mathrm{b},k}^{t+1}). \tag{8}$$

The client then sends the pair (z_k, y_k) to the server, which concatenates all received data from the clients into $Z = \{z_1, z_2, \ldots, z_N\}$ and $Y = \{y_1, y_2, \ldots, y_N\}$, and broadcasts the pair (Z, Y) to all the clients.

2) Modular Block Update: Upon receiving the concatenated data from the server, each client k performs a local update on its modular block parameters $\boldsymbol{\theta}_{\mathrm{m},k}^{t}$, as follows

$$\boldsymbol{\theta}_{\mathrm{m},k}^{t+1} = \boldsymbol{\theta}_{\mathrm{m},k}^{t} - \eta_{\mathrm{m}} \sum_{i=1}^{N} \Delta_{\boldsymbol{\theta}_{\mathrm{m},k}} \ell(f_{\mathrm{m},k}(z_i; \boldsymbol{\theta}_{\mathrm{m},k}^{t}), y_i), \tag{9}$$

where η_m is the learning rate and $\Delta_{\boldsymbol{\theta}_{\mathrm{m},k}} \ell(\cdot)$ is the gradient of the loss with respect to the modular block parameters.

During the inference phase, each client k uses its local model to generate predictions $\hat{y}_k$ as follows

$$\hat{y}_k = f_{\mathrm{m},k}(f_{\mathrm{b},k}(x_k; \boldsymbol{\theta}_{\mathrm{b},k}); \boldsymbol{\theta}_{\mathrm{m},k}). \tag{10}$$

Since modular blocks from any client can be composed with the base block of client k, inter-operable inference can be performed as

$$\hat{y}_{k,i} = f_{\mathrm{m},i}(f_{\mathrm{b},k}(x_k; \boldsymbol{\theta}_{\mathrm{b},k}); \boldsymbol{\theta}_{\mathrm{m},i}), \quad \forall i = \{1, 2, \ldots, N\}, \tag{11}$$

where $\hat{y}_{k,i}$ denotes the prediction for client k using its own base block combined with the modular block of client i. Table I summarizes the key differences between our proposed IFL architecture and existing schemes such as FL and FSL.

IV. NUMERICAL RESULTS

TABLE II

CLIENT-SPECIFIC MODEL ARCHITECTURES USED IN THE IFL FRAMEWORK. THE FUSION-LAYER IS HIGHLIGHTED IN BOLD

Client	**Base Block Layers**	**Modular Block Layers**
1	Conv(1,16) $\rightarrow$ Conv(16,32) $\rightarrow$ **Conv(32,48)**	FC(432,256) $\rightarrow$ FC(256,128) $\rightarrow$ FC(128,64) $\rightarrow$ FC(64,10)
2	Conv(1,16) $\rightarrow$ Conv(16,32) $\rightarrow$ **FC(1568,432)**	FC(432,128) $\rightarrow$ FC(128,10)
3	**FC(784,432)**	FC(432,256) $\rightarrow$ FC(256,128) $\rightarrow$ FC(128,64) $\rightarrow$ FC(64,10)
4	FC(784,1024) $\rightarrow$ FC(1024,512) $\rightarrow$ **FC(512,432)**	FC(432,10)

A. Experimental Setup

We consider a collaborative training setup with $N = 4$ clients and a central server. Each client performs $\tau = 10$ local iterations to update its base block before communicating with the server. The training is carried out for $T = 200$ communication rounds. We report results averaged over 10 Monte Carlo runs.

1) Baselines: Our proposed algorithm IFL is compared to the following baselines.

- FL (Federated Learning): Clients train their local models on local data and transmit the updated models to the server, which aggregates them using FedAvg [4], then sends the updated global model back to the clients.
- FSL (Federated Split Learning): Based on [9], clients share cut-layer outputs with the server. The server-side model updates are averaged, while client-side models remain personalized to local data.

2) Dataset: We consider an image classification task using Kuzushiji-MNIST dataset, where we have $50k$ training samples and $10k$ test samples [16]. To measure the effect of data heterogeneity, we generate Non-Independent and Identically Distributed (Non-IID) data using a Dirichlet distribution with concentration parameter α, which controls the per-class sample distribution among clients [17]. We use the following configuration parameters: α = 0.5, $\eta_b = 0.01$, $\eta_m = 0.01$, $B = 32$, and $z_k \in \mathbb{R}^{B\times 432}, \quad \forall k \in \{1, 2, \ldots, N\}$.

3) Models: To validate collaborative learning with heterogeneous models, each client is assigned a different architecture, reflecting vendor-specific implementations. Details of the models

are summarized in Table II. Convolutional layers (Conv) are followed by pooling layers, while fully connected (FC) layers are followed by ReLU activations, except for the output layer. Importantly, the fusion-layer type may differ across clients, e.g., client 1 may use Conv-based fusion-layer, while others use FC-based layers. This highlights the importance of a standardized fusion-layer output dimension, without requiring identical fusion-layer types.

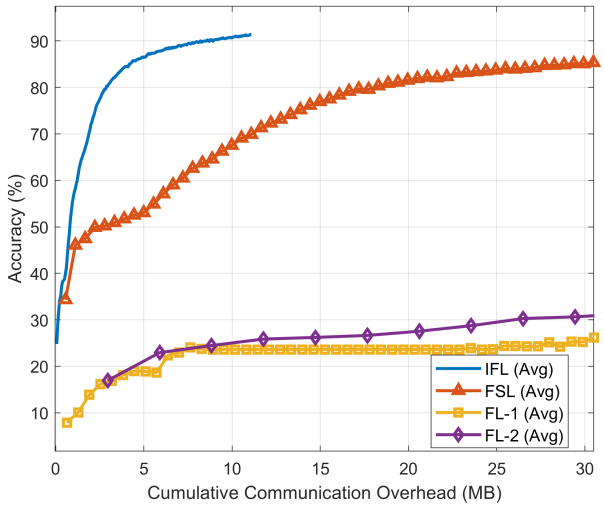

Fig. 2. Test accuracy averaged over all clients versus cumulative communication overhead (MB) for IFL (proposed), FSL, and FL variants.

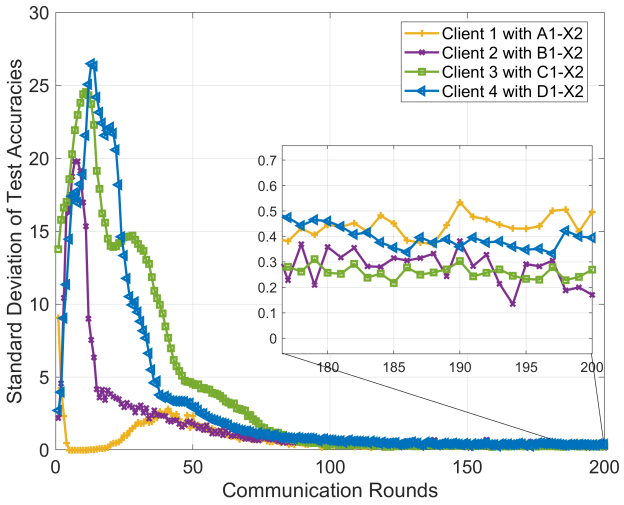

Fig. 3. Standard deviation of test accuracy for each client's base block combined with modular blocks from all clients over communication rounds.

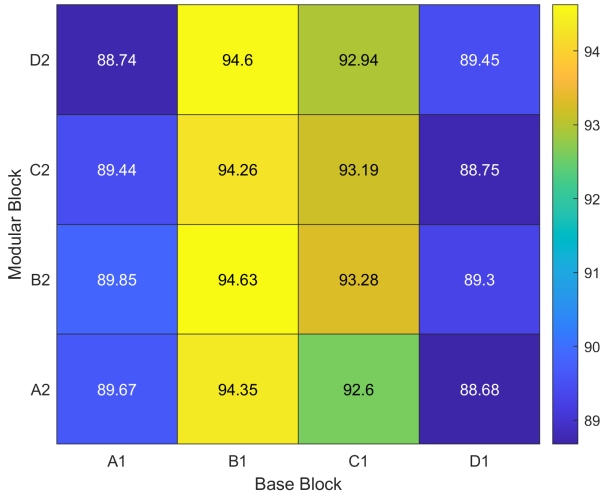

Fig. 4. Test accuracy matrix for all base blocks and modular blocks combinations across clients.

B. Results and Discussion

1) Communication Efficiency: Fig. 2 compares test accuracy versus cumulative communication overhead (in MB) incurred during training for IFL, FSL, and two FL variants. FL-1 deploys the smallest model (from client 1) across all clients, whereas FL-2 adopts the more accurate but larger model (from client 2). IFL achieves 90% test accuracy with only 8.5 MB of uplink data, while FSL reaches just 64% at the same communication cost. This demonstrates the advantage of IFL's multiple local updates in reducing transmission frequency. Both FL variants incur high overhead due to transmitting full models in every communication round.

2) Training with Heterogeneous Models: Fig. 3 presents the standard deviation (SD) of the test accuracy for each client's base block when paired with the modular blocks of all other clients over communication rounds. For example, the label $A1$–$X2$ denotes that client 1's base block ($A1$) is composed with all modular blocks from all the clients. At the end of the training, all SD values fell below 0.6, indicating consistent performance across heterogeneous combinations. This robustness was due to the fact that all modular blocks were trained on the same data sent by the server, which facilitated convergence despite architectural differences. These results confirmed IFL's effectiveness in supporting cross-vendor collaborative learning.

3) Cross-Vendor Interoperable Inference: We evaluate inference interoperability by constructing an accuracy matrix, where each entry shows the test accuracy from combining a client's base block with another client's modular block. Fig. 4 shows that cross-client combinations achieve

performance that is comparable to, and in some cases even exceed, that of local compositions (for instance, $A1$-$B2$ outperforms $A1$-$A2$, and $C1$-$B2$ surpasses $C1$-$C2$). This demonstrates the generalization and composability of the proposed framework.

V. Conclusion

We presented Interoperable Federated Learning (IFL), a communication-efficient framework for collaborative learning across clients with heterogeneous model architectures. By standardizing the fusion-layer output and decoupling model components, IFL enables cross-vendor training and modular inference without exposing model parameters or architectures. Experimental results demonstrate that IFL maintains robust performance across diverse model architectures and enables the seamless interchange of modular blocks across clients, highlighting its potential for scalable, interoperable deployment in multi-vendor AI systems.

References

[1] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[2] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten, "The road towards 6g: A comprehensive survey," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334–366, 2021.

[3] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, "6g wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 28–41, 2019.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[5] Z. Yang, M. Chen, K.-K. Wong, H. V. Poor, and S. Cui, "Federated learning for 6g: Applications, challenges, and opportunities," *Engineering*, vol. 8, pp. 33–41, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095809921005245

[6] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021. [Online]. Available: http://dx.doi.org/10.1561/2200000083

[7] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018. [Online]. Available: https://arxiv.org/abs/1812.00564

[8] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804518301590

[9] M. Kim, A. DeRieux, and W. Saad, "A bargaining game for personalized, energy efficient split learning over wireless networks," in *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, 2023, pp. 1–6.

[10] M. Krouka, A. Elgabli, C. b. Issaid, and M. Bennis, "Communication-efficient split learning based on analog communication and over the air aggregation," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[11] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," 2022. [Online]. Available: https://arxiv.org/abs/2004.12088

[12] W. Fan, J. Yoon, X. Li, H. Shao, and B. Ji, "P3sl: Personalized privacy-preserving split learning on heterogeneous edge devices," 2025. [Online]. Available: https://arxiv.org/abs/2507.17228

[13] Y. Papageorgiou, Y. Thomas, A. Filippakopoulos, R. Khalili, and I. Koutsopoulos, "Collaborative split federated learning with parallel training and aggregation," 2025. [Online]. Available: https://arxiv.org/abs/2504.15724

[14] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[15] S. Ruder, "An overview of gradient descent optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1609.04747

[16] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," 2018. [Online]. Available: https://arxiv.org/abs/1812.01718

[17] I. Moreno-Sánchez, F. Font-Clos, and A. Corral, "Large-scale analysis of zipf's law in english texts," *PLOS ONE*, vol. 11, pp. 1–19, 01 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0147073