Split Learning-Enabled Framework for Secure and Lightweight Internet of Medical Things Systems

Siva Sai, Manish Prasad, Animesh Bhargava, Vinay Chamola *Senior Member, IEEE*, Rajkumar Buyya *Fellow, IEEE*

***Abstract*—The rapid growth of Internet of Medical Things (IoMT) devices has resulted in significant security risks, particularly the risk of malware attacks on resource-constrained devices. Conventional deep learning methods are impractical due to resource limitations, while Federated Learning (FL) suffers from high communication overhead and vulnerability to non-IID (heterogeneous) data. In this paper, we propose a split learning (SL) based framework for IoMT malware detection through image-based classification. By dividing the neural network training between the clients and an edge server, the framework reduces computational burden on resource-constrained clients while ensuring data privacy. We formulate a joint optimization problem that balances computation cost and communication efficiency by using a game-theoretic approach for attaining better training performance. Experimental evaluations show that the proposed framework outperforms popular FL methods in terms of accuracy, F1-score, high convergence speed, and less resource consumption. These results establish the potential of SL as a scalable and secure paradigm for next-generation IoMT security.**

***Index Terms*—Split Learning, Federated Learning, Distributed Machine Learning, Joint Optimization, Game Theory, IoMT Security**

I. Introduction

Internet of Things (IoT) has emerged as a popular paradigm for connecting vast networks of computing devices, sensors, software, and many more, with enhanced communicative capabilities, automation, and efficiency, thus revolutionizing both industrial and commercial use cases. However, this rapid proliferation of IoT devices has also created many security challenges [1]–[3]. Internet of Medical Things (IoMT) malware [4] is a type of malicious software specifically designed to target IoMT devices. Unlike traditional computer viruses, IoMT malware often operates on devices with limited processing power and memory, making it more difficult to detect and remove. Detection of IoMT malware using image-based techniques has emerged as a promising approach in cybersecurity [5], [6]. This method leverages visual representations of malware files to identify and classify malicious software targeting IoMT devices. By converting malware executable files into grayscale images, unique texture patterns and structural characteristics of the malware code are captured, which are then used as input for specially designed deep Convolutional Neural Networks (CNNs) models, which have proven to be highly effective in extracting discriminative features from these malware images at different abstraction levels; achieving accuracy rates of up to 90% for constrained input sizes, and exceeding 95% when higher-dimensional representations and heavier models are used [7]. As opposed to regular deep learning techniques, distributed learning methods are much better suited for IoMT devices since these devices often have limited processing power, memory, and energy [8], [9], and distributed learning [10], [11] methods make better use of all available resources by allowing the computational load to be effectively spread across multiple devices [12]–[14]. Federated Learning (FL) [15], [16] is traditionally used in training IoMT-based models [17]. In the context of IoMT malware detection, FL enables devices to train detection models collaboratively while preserving data privacy and confidentiality. This approach addresses scalability issues inherent in traditional centralized machine learning methods and reduces communication overhead, which is particularly beneficial in resource-constrained IoMT environments [18]. However, FL faces several limitations, of which the high computational burden on the resource-poor clients is of great concern in IoMT scenarios.

Split Learning (SL) has emerged as a popular alternative in the domain [23], offering several benefits over traditional FL in terms of security and performance. SL allows for more flexible model architectures that can adapt to different privacy requirements and computational constraints of IoMT devices. By partitioning the network between layers, SL can offload part of the training process to more powerful servers, alleviating the computational burden on resource-constrained IoMT devices. Furthermore, SL has shown superior performance in specific tasks, which could lead to more efficient training of IoMT malware detection models. For instance, SL frameworks have accelerated training and improved convergence rates in environmental monitoring scenarios, where scalable and energy-efficient models are critical for deploying water quality and pollution monitoring systems [24], [25]. SL-based frameworks especially excel in areas where edge nodes operate under limited power supply and are deployed in remote areas, such as forests or hilly regions, for monitoring purposes [26], [27].

Siva Sai is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore (e-mail: siva.sai@nus.edu.sg).

Manish Prasad and Vinay Chamola are with the Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani Campus, India 333031 (e-mails: {p20240903, vinay.chamola}@pilani.bits-pilani.ac.in).

Animesh Bhargava is with the Department of Computer Science and Information Systems, BITS-Pilani, Pilani Campus, India 333031 (e-mail: h20190545@pilani.bits-pilani.ac.in).

Vinay Chamola is also with APPCAIR, BITS-Pilani, Pilani campus.

Rajkumar Buyya is with the Quantum Cloud Computing and Distributed Systems (qCLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia (e-mail: rbuyya@unimelb.edu.au).

Table I: Qualitative comparison with related works. "Image-based" indicates malware-to-image representation. "Comm.-efficient" indicates communication-awareness/optimization. "Client compute-aware" indicates explicit client-side compute reduction/constraints.

Work	Key Contribution	SL	Image based	Comm. eff.	Client compute	Joint opt.
Asam et al. [19]	Modular CNN for IoMT malware images	×	✓	×	×	×
Sanchez et al. [20]	FL with MLP/AE; centralized-level accuracy	×	×	×	×	×
Li et al. [21]	Multi-institution SL; 41% client compute reduction	✓	◐	×	✓	×
Singh et al. [22]	SL efficiency study	✓	✓	◐	◐	×
Proposed Work	SL for IoMT malware with game-theoretic cost–communication balance	✓	✓	✓	✓	✓

Legend: ✓ present, ◐ partial, × not present

In this paper, we propose a novel approach that applies SL techniques to enhance IoMT malware detection focusing on IoMT [28] devices using image-based classification. Our method takes advantage of SL to distribute the computational load between IoMT devices and edge servers, using joint optimizations for computation cost for edge devices and communications efficiency between client and server, while maintaining data privacy. By combining image-based IoMT malware detection with the proposed SL framework, we aim to develop a more secure, efficient, and privacy-preserving system to protect IoMT devices against evolving malware threats. This approach holds promise in addressing the unique challenges posed by the diverse and resource-constrained nature of IoMT environments.

The rest of the paper is organized as follows. Section II describes the works related to the proposed model. In Section III, we describe the proposed methodology, followed by a discussion on performance evaluation in Section IV. Finally, we conclude the paper in Section V.

II. Related Work

A. IoMT Malware Detection

Asam et al. [19] proposed the iMDA architecture, which incorporates a modular design featuring edge exploration, multipath dilated convolution, and channel squeezing/boosting in CNN. The architecture was evaluated using a benchmark IoMT dataset with image-based malware representations. Their study was limited to static image-based analysis and focused only on ARM-based IoMT malware. Rey et al. [20] developed a Federated Learning framework using both supervised (multi-layer perceptron) and unsupervised (autoencoder) models and achieved similar accuracy to centralized models (99% detection rate). However, the model is highly vulnerable to adversarial attacks. Babbar et al. [29] proposed a Federated Learning Framework using FedAvg aggregation for distributed attack detection across IoMT domains and validated using the BoT-IoMT dataset with multiple attack categories (DDoS, reconnaissance) [30], [31]. They achieved 99.9% accuracy after 30 iterations with minimal network delay and memory usage, but faced high communication overhead in large-scale deployments. Asiri et al. [32] introduce a federated learning framework with privacy and reliability enhancements for IoMT malware detection by using elliptic curve digital signatures (ECDSA) to verify clients and homomorphic encryption (HE) to protect local model weights and integrating blockchain-based smart contracts [33] for decentralized client evaluation and aggregator failure tracking.

B. Split Learning

Li et al. [21] tested a multi-institutional SL framework on five medical datasets. They compared SL and FL for EHR and imaging data and obtained 96% agreement with centralized models. However, the model was limited to 3-way model splits and incurred high communication latency (around 120ms/round). Singh et al. [22] compared the communication efficiencies of split learning and federated learning. The authors demonstrated that the split learning should be favored as the model sizes and the number of clients grow. On the other hand, federated learning becomes more communication-efficient when the number of data samples is limited while keeping the model sizes and number of clients low [34], [35]. A major drawback of this work is that it fails to analyze resource utilization (the number of epochs required for convergence under various scenarios). Table I compares the proposed model with the related works.

III. Proposed Methodology

In this section, we describe the techniques and methodologies we adopt for implementing the proposed split learning framework (see Figure 1), including the overall workflow, SL algorithms, and optimization techniques.

A. Overall Workflow

Initially, IoMT devices and edge servers that want to participate in collaborative training are registered in the system. The split learning paradigm strategically divides the custom Convolutional Neural Network (CNN) between the server and the clients at a predefined cut layer as per the device constraints. To ensure a better distribution of resources, prevent a single point of failure, and improve reliability, the system incorporates dynamic server allocation. Once selected, the server initiates the global model and distributes the client-side

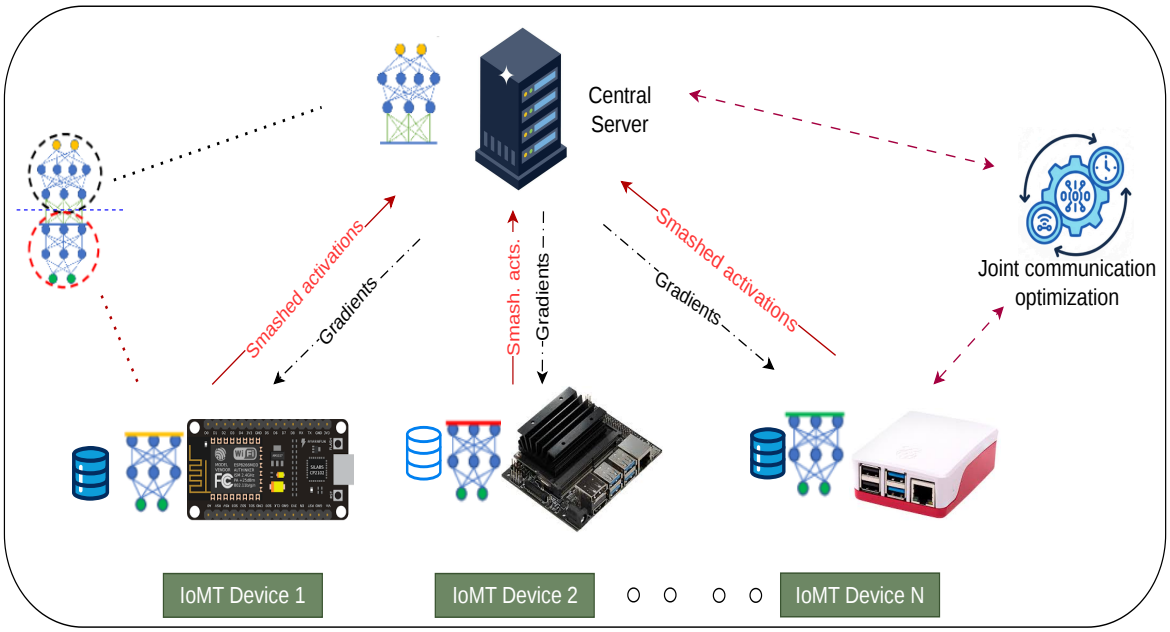


Figure 1: Proposed split learning (SL) framework for IoMT malware detection. Clients execute layers up to the cut layer and transmit smashed activations to the server; the server completes forward/backward propagation and returns boundary gradients. A joint resource allocator optimizes bandwidth/power and compute to meet an energy–latency objective under system constraints.

portions of the neural network to all participating IoMT devices.

Each training round involves the clients processing their local image-based malware datasets through their portion of the network and transmitting the resulting activations, known as smashed activations, to the server. The server continues the forward pass, computes the loss, and performs backpropagation. The newly computed gradients are then sent to the client-side layers and update their weights accordingly. This process continues to iterate until a predefined stopping condition is reached.

The framework uses a joint strategy for computation and communication efficiency to optimize performance for resource-constrained IoMT devices. Depending on available device resources, cut-layer portions within the CNN are chosen, aiming to minimize client-side computational costs and bandwidth usage. Using stratified sampling and balanced train-validation-test splits improves model generalization despite heterogeneous local datasets. Metrics, including training loss, accuracy, F1 score, processing time, and total computation (TFLOPs), are tracked to compare the split learning approach with traditional federated learning.

B. Learning Framework

Split learning [36] has emerged as a distributed and collaborative model training paradigm that addresses many of the issues encountered in federated learning. Algorithm 2 shows how the model computations take place, including the data transfer between clients and server [37]. The model training process in SL involves several steps that are listed below:

1) Each client processes its private data using its assigned layers of the model, i.e., up to the cut layer. The resulting activations, also called *smashed activations*, are securely transmitted to the server.
2) Upon receiving the activations, the server continues the forward pass through the remaining layers of the neural network, generating predictions and computing the training loss.
3) The server performs backpropagation on its portion of the network, obtaining gradients for both its own layers and for the cut layer.
4) The boundary gradients are sent back to the clients. Using these, each client executes backpropagation on its local layers and updates its model parameters.
5) Steps 1–4 are repeated collaboratively across all participating clients for several training rounds. The cycle continues until a convergence condition (e.g., a predefined number of epochs, stable loss, or target accuracy) is met.

C. Joint optimization for computation cost and communication latency

To increase the model's performance on IoMT edge devices, we adopt a game-theoretic approach to formulate a mathematical model for the joint optimization of computation cost and communication efficiency with minimal compromise in training accuracy. By adapting and extending established formulations from federated learning [38] and mobile edge computing, we model a unified optimization problem that formalizes the trade-offs between energy, latency, and accuracy [39], [40] in the proposed split learning framework. We consider split learning (SL) over a wireless star topology with

a central server and M IoMT devices (clients), indexed by the set $\mathcal{U} \triangleq \{1, \ldots, M\}$.

Each device $u \in \mathcal{U}$ executes the client-side sub-network up to the split layer and transmits the corresponding smashed activations to the server. We assume frequency-division multiple access (FDMA) for the uplink. As the server's downlink power is typically much larger than client uplink power, downlink latency is negligible and thus ignored.

Let $\mathcal{D}_u = \bigcup_{i=1}^{D_u} \{\mathbf{x}_{ui}, y_{ui}\}$ denote device u's dataset. The SL training aims to minimize the global empirical loss:

$$\min_{\mathbf{w} \in \mathbb{R}^p} \; F(\mathbf{w}) = \sum_{u=1}^{M} \frac{D_u}{\sum_{j=1}^{M} D_j} \, \ell_u(\mathbf{w}), \tag{1}$$

$$\ell_u(\mathbf{w}) = \frac{1}{D_u} \sum_{i=1}^{D_u} f_{ui}(\mathbf{w}), \tag{2}$$

where $f_{ui}(\cdot)$ is the sample loss.

1) Uplink Rate, Energy, and Time: Under FDMA, the achievable uplink rate of device u is

$$\varrho_u \;=\; b_u \log_2\!\left(1 + \frac{\rho_u \gamma_u}{\sigma^2 b_u}\right), \tag{3}$$

where b_u is bandwidth, ρ_u is transmit power, γ_u is channel gain, and σ^2 is noise power. Let δ_u (bits) be the total uplink payload per round. The uplink time and transmission energy per round are:

$$t_u^{ul} = \frac{\delta_u}{\varrho_u}, \tag{4}$$

$$\epsilon_u^{tx} = \rho_u \, t_u^{ul} \tag{5}$$

2) Local Computation Energy and Time: Let ϕ_u be the CPU frequency, χ_u the cycles per sample, and ξ the switched capacitance. With L local iterations, the local computation energy and time per round are:

$$\epsilon_u^{\mathrm{cmp}} = \xi \, L \, \chi_u D_u \phi_u^2, \qquad t_u^{\mathrm{cmp}} = L \, \frac{\chi_u D_u}{\phi_u}. \tag{6}$$

Over G global rounds, total energy $\mathcal{E}$ and completion time T are:

$$\mathcal{E} = G \sum_{u=1}^{M} \left(\epsilon_u^{\mathrm{tx}} + \epsilon_u^{\mathrm{cmp}}\right), \tag{7}$$

$$T = G \max_{u \in \mathcal{U}} \left\{ t_u^{\mathrm{cmp}} + t_u^{\mathrm{ul}} \right\}. \tag{8}$$

3) Optimization Problem: We jointly allocate transmit powers $\{\rho_u\}$, bandwidths $\{b_u\}$ and CPU frequencies $\{\phi_u\}$ to balance energy and time, using a weight $\alpha \in [0, 1]$.

$$\min_{\{\rho_u, \phi_u, b_u\},\; \vartheta} \quad \alpha G \sum_{u=1}^{M} \left(\rho_u \frac{\delta_u}{\varrho_u} + \xi L \chi_u D_u \phi_u^2 \right) + (1-\alpha) G \vartheta \tag{9a}$$

$$\text{s.t.} \quad t_u^{\mathrm{cmp}} + t_u^{\mathrm{ul}} \le \vartheta, \quad \forall u \in \mathcal{U}, \tag{9b}$$

$$\rho_u^{\min} \le \rho_u \le \rho_u^{\max}, \quad \phi_u^{\min} \le \phi_u \le \phi_u^{\max}, \quad \forall u, \tag{9c}$$

$$\sum_{u=1}^{M} b_u \le \bar{b}, \quad b_u \ge 0, \quad \forall u, \tag{9d}$$

Algorithm 1 Resource Allocation for SL (Energy–Latency Trade-off)

Input: Initial feasible $(\rho^{(0)}, b^{(0)}, \phi^{(0)})$; system parameters; bounds on (ρ, b, ϕ); tolerances $\epsilon_{\text{in}}, \epsilon_{\text{out}}$
Output: Optimized allocations $(\rho^\star, b^\star, \phi^\star)$ and round latency $\vartheta^\star$

$k \leftarrow 0$
repeat
  ▷ **Subproblem A: CPU frequency update and tight latency**
  $\varrho \leftarrow \text{ComputeRates}(\rho^{(k)}, b^{(k)})$ ▷ rate function $\mathcal{G}_u(\cdot)$
  $\ell^\star \leftarrow \text{SolveDualSubproblemA}(\varrho)$
  $\phi^{(k+1)} \leftarrow \text{UpdateCPUFrequency}(\ell^\star)$ ▷ projection/clipping to $[\phi^{\min}, \phi^{\max}]$
  $\vartheta^{(k+1)} \leftarrow \text{ComputeTightLatency}(\varrho, \phi^{(k+1)})$
  ▷ **Subproblem B: Bandwidth and power update (inner loop)**
  $\underline{\varrho} \leftarrow \text{ComputeMinRate}(\vartheta^{(k+1)}, \phi^{(k+1)})$
  $(\upsilon, \zeta) \leftarrow \text{InitAuxVars}(\rho^{(k)}, b^{(k)})$
  repeat
    $(\rho, b) \leftarrow \text{SolveSubproblemB\_KKT}(\underline{\varrho}, \phi^{(k+1)}, \vartheta^{(k+1)}, \upsilon, \zeta)$
    $(\upsilon, \zeta) \leftarrow \text{UpdateAuxVars}(\rho, b)$
  until $\text{Residual}(\upsilon, \zeta) \le \epsilon_{\text{in}}$
  $(\rho^{(k+1)}, b^{(k+1)}) \leftarrow (\rho, b)$
  $k \leftarrow k + 1$
until $\text{Converged}(\rho^{(k)}, b^{(k)}, \phi^{(k)})$ is true under ϵ_{out}
return $(\rho^\star, b^\star, \phi^\star, \vartheta^\star) \leftarrow (\rho^{(k)}, b^{(k)}, \phi^{(k)}, \vartheta^{(k)})$

where ϑ is an auxiliary variable for per-round latency. Constraint (9b) is equivalent to:

$$\varrho_u \;\ge\; \underline{\varrho}_u \triangleq \frac{\delta_u}{\vartheta - \frac{L \chi_u D_u}{\phi_u}}, \qquad \forall u \in \mathcal{U}. \tag{10}$$

Problem (9) is non-convex, so we use problem decomposition.

4) Problem Decomposition: We separate (9) into (i) frequency–latency allocation over $(\{\phi_u\}, \vartheta)$ and (ii) radio allocation over $(\{\rho_u, b_u\})$.

a) Subproblem A (CPU & latency).: For fixed ϱ_u, this subproblem is convex.

$$\min_{\{\phi_u\}, \vartheta} \quad \alpha G \sum_{u=1}^{M} \xi L \chi_u D_u \phi_u^2 + (1-\alpha) G \vartheta \tag{11a}$$

$$\text{s.t.} \quad L \frac{\chi_u D_u}{\phi_u} + \frac{\delta_u}{\varrho_u} \le \vartheta, \quad \phi_u^{\min} \le \phi_u \le \phi_u^{\max}, \quad \forall u. \tag{11b}$$

Applying KKT conditions yields the optimal CPU frequency:

$$\phi_u^\star = \sqrt[3]{\frac{\ell_u}{2 \alpha G \xi}}, \tag{12}$$

where ℓ_u are Lagrange multipliers. The dual problem is a convex resource-splitting program:

$$\max_{\ell_u \ge 0} \quad \sum_{u=1}^{M} \left((2^{-\frac{2}{3}} + 2^{\frac{1}{3}}) \, L (\alpha \xi G)^{\frac{1}{3}} \chi_u D_u \, \ell_u^{\frac{2}{3}} + \tfrac{\delta_u}{\varrho_u} \ell_u \right) \tag{13a}$$

$$\text{s.t.} \quad \sum_{u=1}^{M} \ell_u = (1-\alpha) G. \tag{13b}$$

b) Subproblem B (power & bandwidth).: Define $\mathcal{G}_u(\rho_u, b_u) \triangleq b_u \log_2(1 + \frac{\rho_u \gamma_u}{\sigma^2 b_u})$. The radio subproblem is:

$$\min_{\{\rho_u, b_u, \zeta_u\}} \quad \alpha G \sum_{u=1}^{M} \zeta_u \tag{14a}$$

$$\text{s.t.} \quad \rho_u^{\min} \le \rho_u \le \rho_u^{\max}, \quad \sum_{u=1}^{M} b_u \le \bar{b}, \tag{14b}$$

$$\mathcal{G}_u(\rho_u, b_u) \ge \underline{\varrho}_u, \quad \rho_u \delta_u - \zeta_u \mathcal{G}_u(\rho_u, b_u) \le 0, \forall u. \tag{14c}$$

This can be solved via an equivalent subtractive-form program by finding $\{\upsilon_u^\star\}$ such that solving for fixed (υ_u, ζ_u):

$$\min_{\{\rho_u, b_u\}} \quad \sum_{u=1}^{M} \upsilon_u \big(\rho_u \delta_u - \zeta_u \mathcal{G}_u(\rho_u, b_u)\big) \tag{15a}$$

$$\text{s.t.} \quad \rho_u^{\min} \le \rho_u \le \rho_u^{\max}, \quad \sum b_u \le \bar{b}, \quad \mathcal{G}_u \ge \underline{\varrho}_u, \tag{15b}$$

and then updating $\upsilon_u \leftarrow \frac{\alpha G}{\mathcal{G}_u(\rho_u, b_u)}$ and $\zeta_u \leftarrow \frac{\rho_u \delta_u}{\mathcal{G}_u(\rho_u, b_u)}$ drives the solution. The KKT conditions for (15) yield closed-form characterizations. Let $\omega_u \triangleq \frac{\rho_u \gamma_u}{\sigma^2 b_u}$. Optimality implies:

$$\upsilon_u \left(\delta_u - \frac{\zeta_u \gamma_u}{\sigma^2 (1+\omega_u) \ln 2} \right) - \frac{\theta_u \gamma_u}{\sigma^2 (1+\omega_u) \ln 2} = 0, \tag{16}$$

$$- (\upsilon_u \zeta_u + \theta_u) \log_2(1+\omega_u)$$

$$+ \frac{(\upsilon_u \zeta_u + \theta_u) \rho_u \gamma_u}{(1+\omega_u) \ln 2 \, \sigma^2 b_u} + \mu = 0, \tag{17}$$

$$\theta_u \Big(b_u \log_2(1+\omega_u) - \underline{\varrho}_u \Big) = 0, \tag{18}$$

$$\mu \left(\sum_{j=1}^{M} b_j - \bar{b} \right) = 0. \tag{19}$$

Solving the above yields:

$$b_u^\star = \begin{cases} \dfrac{\underline{\varrho}_u}{\log_2(1+\Lambda_u)}, & \text{if } \theta_u \neq 0, \\ \text{solution of (15) with } \theta_u = 0, & \text{otherwise,} \end{cases} \tag{20}$$

$$\rho_u^\star = \min\big\{\rho_u^{\max},\ \max\big\{\Gamma(b_u),\ \rho_u^{\min}\big\}\big\}, \tag{21}$$

$$\Gamma(b_u) \triangleq \left(\frac{(\upsilon_u \zeta_u + \theta_u)\gamma_u}{\sigma^2 \delta_u \upsilon_u \ln 2} - 1 \right) \frac{\sigma^2 b_u}{\gamma_u}, \tag{22}$$

$$\Lambda_u \triangleq \frac{(\upsilon_u \zeta_u + \theta_u)\gamma_u}{\sigma^2 \delta_u \upsilon_u \ln 2}. \tag{23}$$

D. Alternating Optimization and Convergence Criterion

We solve (9) by alternating between Subproblem A and Subproblem B as shown in Algorithm 1.

In the inner loop, we employ a diagonal Newton step with $\Phi = [\Phi_1^\top, \Phi_2^\top]^\top \in \mathbb{R}^{2M}$,

$$\Phi_1(\zeta) = \big[-\rho_1\delta_1 + \zeta_1 \mathcal{G}_1(\rho_1, b_1),\ \ldots,\ - \rho_M \delta_M + \zeta_M \mathcal{G}_M(\rho_M, b_M)\big]^\top, \tag{24}$$

$$\Phi_2(\upsilon) = \big[-\alpha G + \upsilon_1 \mathcal{G}_1(\rho_1, b_1),\ \ldots,\ - \alpha G + \upsilon_M \mathcal{G}_M(\rho_M, b_M)\big]^\top. \tag{25}$$

whose Jacobians are diagonal with entries $\mathcal{G}_u(\rho_u, b_u)$, ensuring efficient updates. The alternating optimization method guarantees convergence because each subproblem is convex, each update weakly decreases the global objective, and the objective is bounded below by 0.

Algorithm 2 Proposed Split Learning Procedure

Setup: The server initializes the global model M
On the Server ($\mathbb{S}$):
while stopping condition not satisfied **do**
  Collect gradient or activation updates $\mathbb{U}$ from participating clients
  for each client i **do**
    Aggregate local contributions m into the global model M
  end for
  Broadcast the refined global model M' back to all clients
end while
Terminate

On Client i ($\mathbb{C}_i$):
while training not converged **do**
  Compute local gradients $\nabla_{\theta_i} L_i$ using dataset $\mathbb{D}_i$ and parameters θ_i
  Forward the computed gradients $\nabla_{\theta_i} L_i$ to the server
  Update local parameters by incorporating the received global model M'
end while
Terminate

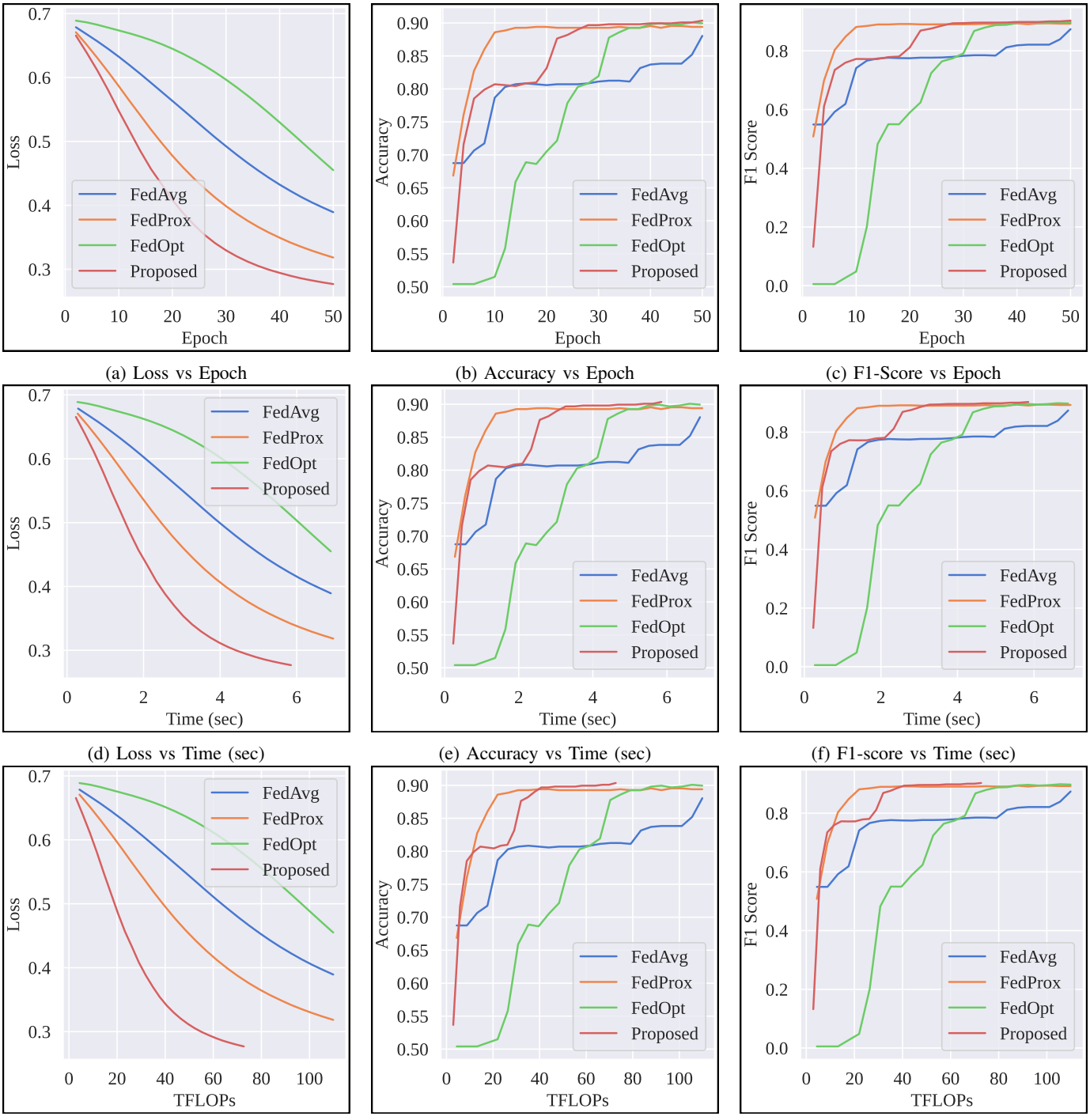
Unlike federated learning (FL), split learning (SL) partitions the network such that clients transmit *smashed activations* rather than full parameter vectors. This distinction is reflected in our formulation through two key parameters: (i) δ_u, representing the SL payload per global round, and (ii) χ_u, capturing the number of client-side CPU cycles per data sample. The optimization problem (9) therefore balances the computation energy ($\propto \phi_u^2$) against communication latency (enforced via $\underline{\varrho}_u$). The weight $\alpha \in [0,1]$ tunes this balance. In energy-limited scenarios, setting $\alpha \to 1$ prioritizes minimizing total energy consumption. Conversely, in latency-critical use cases, setting $\alpha \to 0$ emphasizes minimizing overall completion time. Thus, our formulation provides a flexible framework to adapt SL resource allocation to diverse system objectives and hardware constraints.

IV. Performance Evaluation

In this section, we present several experimental results and analyses related to the proposed model. We also describe the dataset and experimentation tools employed in our work.

Dataset: We have used the IoMT malware image dataset [1] for carrying out the experimental analysis in our work. The dataset contains 17219 images of dimensions 299×299 across three channels. The dataset has a binary classification structure with malware and benign being the two corresponding classes. The dataset is highly imbalanced, with the malware samples comprising only 14.44% of the entire dataset. To deal with this imbalance, we down-sample the majority benign

[1] https://www.kaggle.com/datasets/anaselmasry/iot-malware/data



(a) Loss vs Epoch (b) Accuracy vs Epoch (c) F1-Score vs Epoch

(d) Loss vs Time (sec) (e) Accuracy vs Time (sec) (f) F1-score vs Time (sec)

(g) Loss vs TFLOPs (h) Accuracy vs TFLOPs (i) F1-score vs TFLOPs

Figure 2: Training dynamics comparison between the proposed SL framework and FL baselines (FedAvg, FedProx, FedOpt). Top row: metrics versus epochs. Middle row: metrics versus average client processing time per round (s). Bottom row: metrics versus client compute cost (TFLOPs).

class and apply stratified sampling to ensure a balanced train-validation-test split.

Configuration of ML model: The train-validation-test split we have used is 70-15-15%. After creating the splits, we perform pixel-wise normalization computation for the train split and apply appropriate transforms to architect a rich dataset for training. For the experiments comparing the performance of the proposed SL model and existing techniques, we use a custom-defined Convolution Neural Network (CNN) with 18 hidden layers constructed with the help of PyTorch and related libraries. This gives us a CNN with four sequential blocks with four layers each - Conv2D, ReLU, 0.5-dropout, and half-resolution MaxPool2d, and a fully-connected block with linear, ReLU, and a dropout layer

Table II: Testing performance comparison of the proposed SL framework with FL baselines. Improvements are computed w.r.t. the strongest FL baseline (FedProx).

Method	**Acc (%)**	**Prec (%)**	**Recall (%)**	**F1 (%)**	ΔAcc (pp)	ΔAcc (%)	ΔF1 (pp)	ΔF1 (%)
FedAvg	87.64	93.95	80.38	86.64	-2.03	-2.26	-2.75	-3.08
FedProx	89.67	91.69	87.19	89.39	0.00	0.00	0.00	0.00
FedOpt	89.27	92.60	85.29	88.79	-0.40	-0.45	-0.60	-0.67
Proposed	**90.49**	**92.07**	**88.56**	**90.28**	**+0.82**	**+0.91**	**+0.89**	**+1.00**

Table III: Average client-side training cost per round. Reductions are computed w.r.t. FedAvg (same time/compute as FedOpt in our setup).

Method	**Time (s)** ↓	ΔTime (s)	Time Red. (%)	**TFLOPs** ↓	ΔTFLOPs	Compute Red. (%)
FedAvg	6.89	0.00	0.0	109.87	0.00	0.0
FedProx	6.95	+0.06	-0.9	109.87	0.00	0.0
FedOpt	6.89	0.00	0.0	109.87	0.00	0.0
Proposed	**5.85**	**-1.04**	**15.1**	**72.70**	**-37.17**	**33.8**

for regularization. The convolution layers use an 8-16-32-64 feature maps sequence with kernel, stride, and padding of size 3, 1, 1, respectively. The fully connected linear layer uses 512 features for binary classification. The implementation of the FL algorithm that we use for our experiments uses this custom-defined CNN model for both clients and the server.

Configuration of SL framework: For the proposed split learning implementation, the four sequential blocks provide us with three possible cut layers to split the model at layer positions- 4, 8, and 12, respectively. The client models in this case simulate the decentralized IoMT environments. For the training process, we have used the cross-entropy function for loss computation and the Adam optimizer for gradient descent optimization. Both the FL and SL models are trained for an equivalent of 50 iterations (25 rounds with two epochs for each client) with a batch size of 32 samples per iteration. We use client counts between 2 and 16 for both models to gather a comprehensive report and compare their performance. We track key training parameters for monitoring training progress and performance comparison, including training loss, validation accuracy, F1-score, precision, and recall, processing time for backward and forward passes for client and server, and computation cost in Tera Floating-point Operations (TFLOPs) for each client. We use several Python-based libraries like PyTorch, scikit-learn, numpy, pandas, seaborn, and torchprofile to conduct the experiments. All the experiments in the proposed work are carried out using T4 GPU environments (16GB RAM) on Google Colaboratory[2].

Configuration of joint optimization framework: To validate the mathematical model for resource allocation, we simulated the framework using parameters consistent with established research in wireless federated learning. The simulation environment models a star network consisting of a central base station and 50 devices. These devices are distributed uniformly within a circular area of 500m x 500m. The wireless channel is modeled with a path loss of 128.1 + 37.6 log(distance) (km) and a Gaussian noise power spectral density of -174 dBm/Hz. For the learning process, the number of global aggregation rounds is set to 400, with each round comprising 10 local iterations. Each device is assigned 500 data samples, and the upload data size per device is 28.1 kbits. Device hardware characteristics include a maximum CPU frequency of 2 GHz, a maximum transmission power of 12 dBm, and a total available system bandwidth of 20 MHz. We utilized Python-based numerical optimization library, CVXPY 1.5.2 [3], for the purpose, leveraging the ECOS and SCS solvers for convex and conic constraints. Additionally, we used NumPy for the vectorized computations, SciPy for nonlinear optimization and root-finding, and SymPy for symbolic derivation of analytical gradients and KKT conditions. Iterative convergence was monitored through a residual norm tolerance of 10^{-4}.

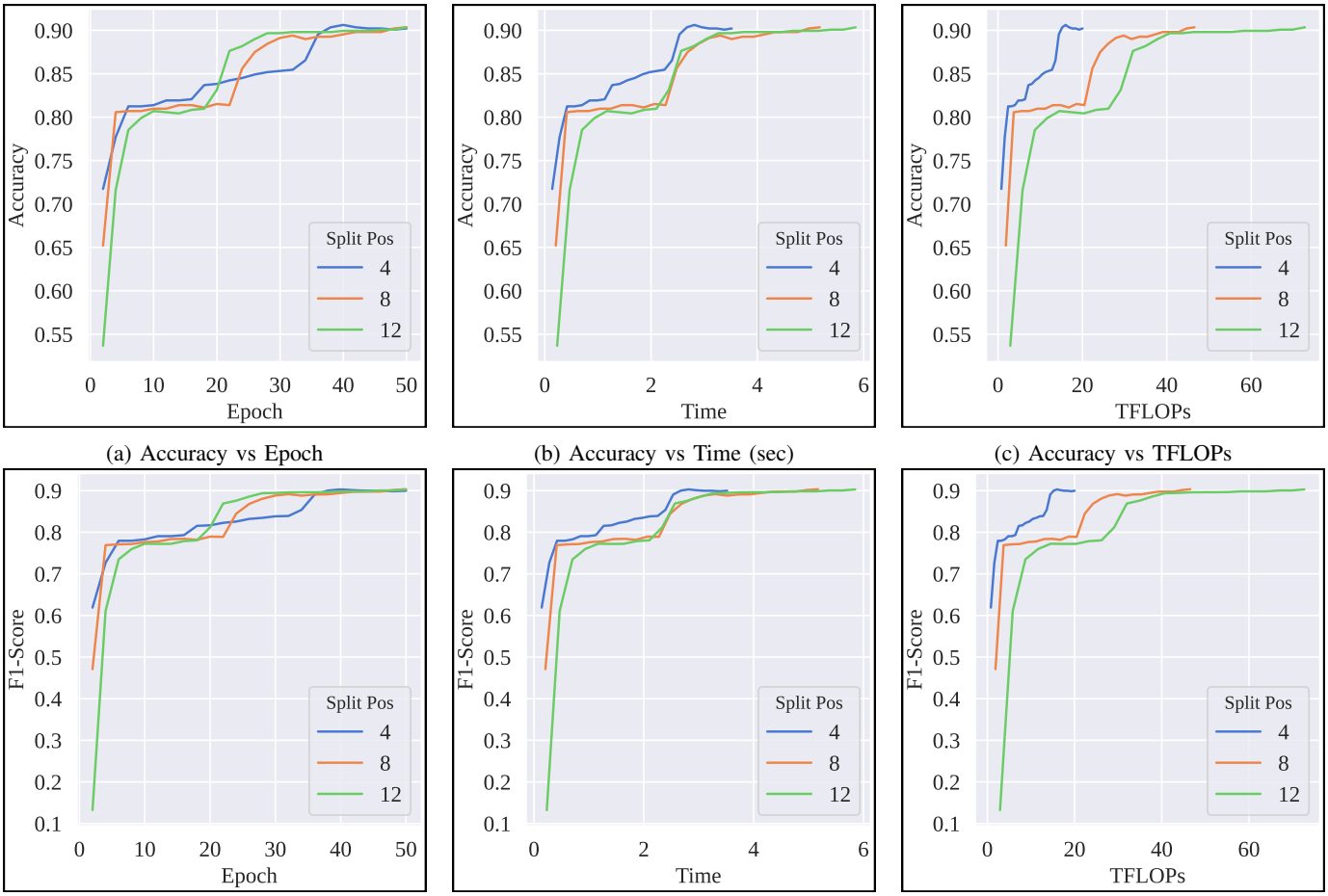
A. Comparing proposed model with the FL baselines

Figure 2 compares the training performance of the proposed SL model with the baseline FL models, namely FedAvg [41], FedProx [42], and FedOpt [43], across 50 epochs. The results are shown for federated models with four clients (FdA-4, FdP-4, FdO-4) and SL with four clients and cut position 1, 12th layer (SpL-4-1). The number of iterations is represented on the X-axis, while the corresponding performance metric is represented on the Y-axis. Figure 2a shows the average training loss across clients for all the federated and split models, while Figures 2b and 2c showcase the validation accuracy and F1-scores at the end of each round for all the models. Figure 2a shows that the proposed SpL-4-1 model is much better at fitting the training data in a decentralized way than federated models, both in terms of speed and quality. The convergence speeds follow the trend: SpL-4-1 $>$ FdP-4 $>$ FdA-4 $>$ FdO-4

Figure 2b indicates that SpL-4-1 consistently outperforms FdA-4 in terms of accuracy as well, while the end performance remains similar compared to FdP-4 and FdO-4. While the

[2]https://colab.research.google.com/

[3]https://github.com/cvxpy/cvxpy



(a) Accuracy vs Epoch (b) Accuracy vs Time (sec) (c) Accuracy vs TFLOPs

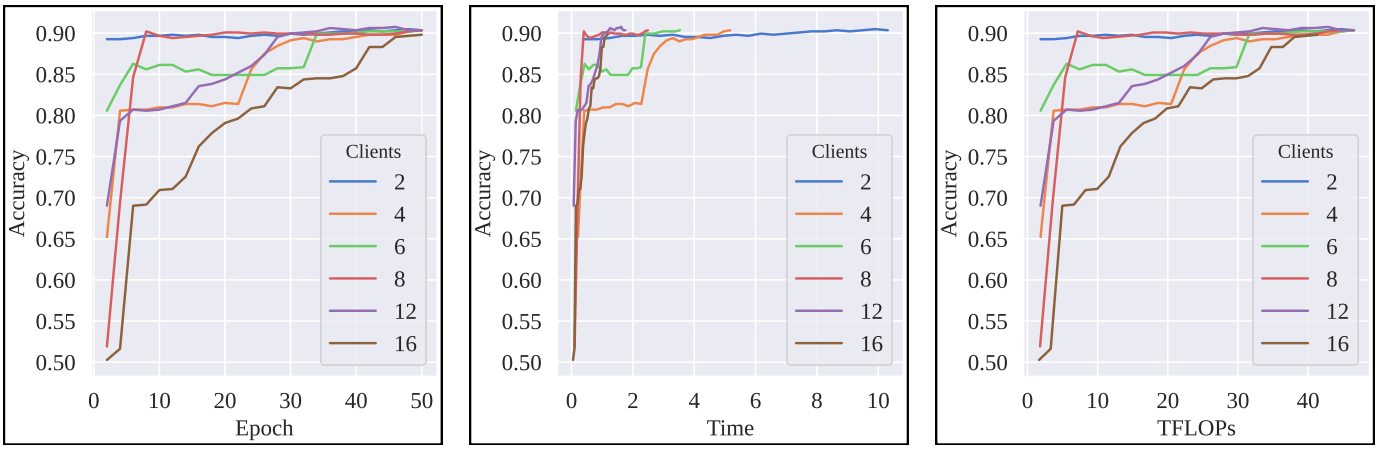
(d) F1-score vs Epoch (e) F1 Score vs Time (sec) (f) F1-score vs TFLOPs

Figure 3: Effect of split position (cut layer) on performance–cost trade-off. Accuracy and F1-score are reported versus epochs, average client processing time per round (s), and client compute (TFLOPs), highlighting how earlier/later cuts shift computation/communication burdens.

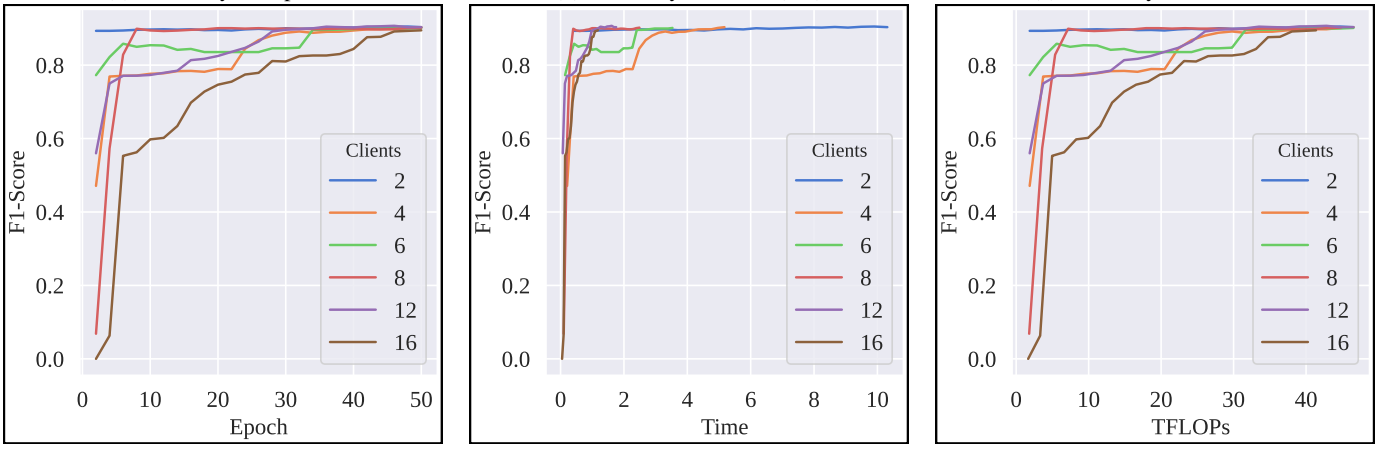
accuracy curve for FdP-4 exhibits a steep initial rise, signifying faster convergence, SpL-4-1 ultimately maintains a rapid and consistent learning trajectory, stabilizing at a higher accuracy level of 90.33% compared to FdA-4, FdP-4, and FdO-4 with 88.04%, 89.4%, and 89.94% accuracy at the end of equivalent epochs, respectively. The F1-score comparison in Figure 2c further highlights these performance dynamics. Notably, FdO-4 exhibits a significant "cold start" period with a zero F1-score, indicating initial difficulty in generalizing. By the final epoch, SpL-4-1, FdP-4, and FdO-4 converge to a similar, high F1-score, significantly outperforming the baseline FdA-4. These results demonstrate that the proposed SL framework enables more effective model training and improved generalization compared to standard FL approaches. Table II presents the test result metrics comparing all the stated models. These results suggest that SL enables more effective model training and improved generalization in a decentralized environment compared to FL.

B. Experiments on Client Communication Efficiency and Computational Load

To compare the impact of the proposed split distributed framework over the federated framework based on computation time and efficiency, we have also measured the time taken in seconds and the computation power units taken in TFLOPS per forward and backward pass for each epoch. From figures 2d, 2e, and 2f, we can see SpL-4-1 demands much lower client processing times for forward and backward passes compared to federated models for training for an equivalent number of epochs. The average forward and backward pass processing time taken by clients for equivalent epochs of training for SpL-4-1 comes out to 5.85 seconds, compared to federated models, which reach almost 7 seconds. Figures 2g, 2h, 2i also show that the computation power required for the convergence from client side is also much lower for SpL-4-1 with a total of 72.70 TFLOPs for all clients (18.175 TFLOPs per client) compared to federated models which takes a significantly higher total of 109.87 TFLOPs from all the clients (27.48 TFLOPs per client), as shown in Table III. From a technical standpoint, the rapid convergence of SL can be attributed to the fact that only a portion of

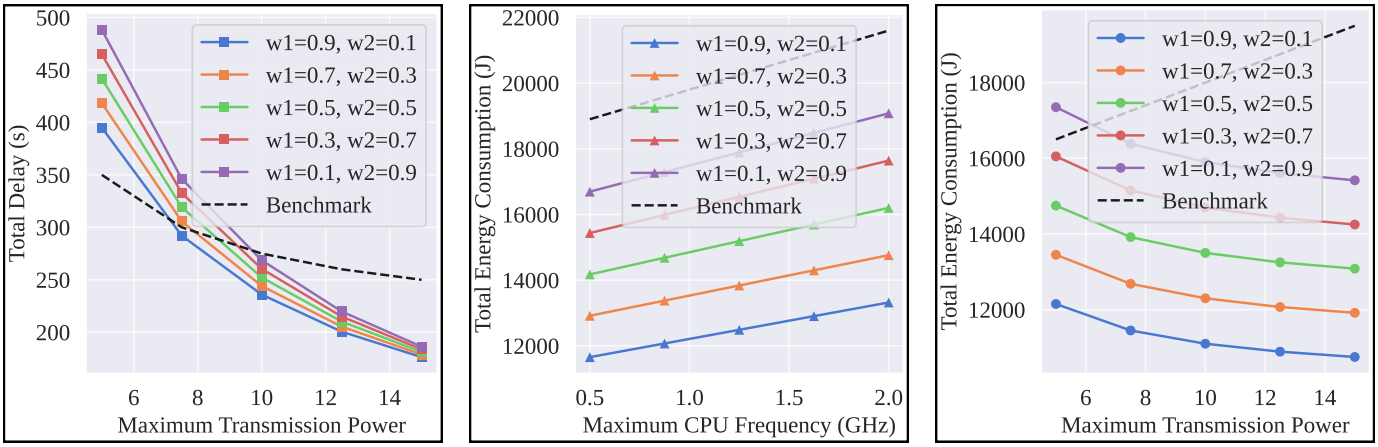


(a) Accuracy vs Epochs (b) Accuracy vs Time (sec) (c) Accuracy vs TFLOPs



(d) F1-score vs Epochs (e) F1-score vs Time (sec) (f) F1-score vs TFLOPs

Figure 4: Effect of the number of participating clients on model performance. Accuracy and F1-score are reported versus epochs, average client processing time per round (s), and client compute (TFLOPs).



(a) Total delay versus maximum transmit power (b) Total energy consumption versus maximum CPU frequency (c) Total energy consumption versus maximum transmit power

Figure 5: Energy–latency trade-offs under different system constraints and weight settings (w_1, w_2).

the model is trained on edge devices while the rest remains on the server. Unlike FL, where the entire model is trained locally before aggregation, SL ensures that intermediate representations are optimized centrally, leading to more stable gradient updates and improved learning efficiency. This better performance and computational efficiency of the split learning algorithm was also noted in previous works [22], [36], [37].

C. Experiments with cut-layer positions

To analyze the impact of the model partitioning strategy on training dynamics, we experimented with three different cut layer positions: 4, 8, and 12, corresponding to the end of the first, second, and third convolutional blocks, respectively, as explained in the model architecture in a previous discussion. The results, visualized in Figure 3, reveal a critical trade-off between client computational load, communication time, and convergence speeds. Figures 3a and 3d show that the cut layer position doesn't impact the performance much. The analysis of processing time (Figures 3b and 3e) also reveals similar trends with the heaviest client-side model (position 12) taking longer processing time to converge. In comparison, the lightest client model (position 4) takes significantly less time. This phenomenon can be attributed to communication overhead: splitting late at position 12 requires transmitting a large, high-dimensional activation tensor from the client, whereas the tensor from position 4 is more spatially compressed, reducing the data transmission bottleneck in each round. Figures 3c and 3f clearly illustrate the relationship between the split position and the client's computational load, measured in TFLOPs. An earlier split at position 4 requires the least computational effort from the client, achieving high performance with approximately 20 TFLOPs, and conversely, splitting at position 12 places the heaviest load on the client, demanding more than 60 TFLOPs to reach convergence. This directly confirms that offloading more layers to the server reduces the client's processing requirements. From a technical standpoint, the choice of the cut layer presents a fundamental trade-off. An early split (e.g., position 4) is ideal for resource-constrained edge devices with limited computational power. However, this comes at the cost of longer processing times at the server. A later split (e.g., position 12) accelerates training in terms of the lesser time required by the server to complete the training process for each client. However, it requires clients with greater computational capacities. Therefore, the optimal cut position is application-dependent, necessitating a balance between the hardware capabilities of client devices and the desired training speed.

D. Experiments with number of clients

In Figure 4, we have visualized the results for our experiments on training the SL model on different numbers of clients. Specifically, we have shown results for training with client counts 2, 4, 6, 8, 12, and 16. Figures 4a, 4b, and 4c show accuracy scores across the different client counts with respect to epochs, average time, and total TFLOPs, respectively. Figures 4d, 4e, and 4f show the F1-scores across the different client counts with respect to epochs, average time, and total TFLOPs. These figures show that increasing the number of clients leads to faster convergence in training for the SL framework. At the same time, the model performance remains approximately the same with a few deviations. The communication overhead for FL increases exponentially with the number of participating clients due to full model weight transmission during each round. In contrast, with SL, we only require transmitting the activations and gradients of a partial model, significantly lowering bandwidth requirements.

E. Numerical Validation of the Joint Optimization Framework

In Figure 5a, we plot the total latency against the maximum power budget. As expected, increasing the transmit power reduces communication delays. This effect is particularly pronounced when the system prioritizes latency minimization (w_2 dominant). Figure 5b investigates the impact of maximum CPU frequency. While higher frequencies reduce computation time, they also incur quadratic energy penalties, reflecting the trade-off between computational efficiency and speed. The figure illustrates how the weighting parameter (w_1, w_2) shifts the balance between these two effects. Figure 5c shows the total energy consumption as a function of the maximum transmit power in normalized units. The proposed allocation strategy shows a consistently lower energy consumption compared with the benchmark scheme, and the gap widens for more energy-focused weight settings.

V. Conclusions and Future Work

In this work, we proposed a split learning framework for IoMT malware detection using image-based representations. The framework strategically distributes computation between edge devices and servers, achieving improved performance in terms of accuracy, convergence speed, and computational efficiency compared to conventional federated learning. Experimental results demonstrate that the proposed approach not only enhances malware detection accuracy but also reduces client-side computation and communication overhead, making it highly suitable for resource-constrained IoMT environments. While the presented framework addresses critical challenges, several avenues remain open for exploration. Future research can extend this work by investigating transfer learning approaches (leveraging pre-trained models to reduce training cost on IoMT devices), vertical split learning (to improve privacy guarantees in multi-party collaborations), and adversarial robustness techniques (to enhance resilience against gradient leakage and poisoning attacks). These directions will further strengthen the applicability of split learning for real-world IoMT security.

Acknowledgments

This work was supported by CHANAKYA Fellowship Program of TIH Foundation for IoMT & IoE (TIH-IoMT) received by Dr. Vinay Chamola under Project Grant File CFP/2022/027 and an ARC Discovery Project.

REFERENCES

[1] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen, and B. Stiller, "Landscape of iot security," *Computer Science Review*, vol. 44, pp. 1–18, 2022.

[2] M. binti Mohamad Noor and W. H. Hassan, "Current research on internet of things (iot) security: A survey," *Computer Networks*, vol. 148, pp. 283–294, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618307035

[3] H. Cao, M. Alrashoud, T. Mohamed, and L. Yang, "An intelligent softwarized resource management and allocation framework for services with personalized intentions in 6g-enabled iot networks," *IEEE Internet of Things Journal*, 2025.

[4] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of iot malware and detection methods based on static features," *ICT express*, vol. 6, no. 4, pp. 280–286, 2020.

[5] B. Tu, T. Zhou, B. Liu, Y. He, J. Li, and A. Plaza, "Multi-scale autoencoder suppression strategy for hyperspectral image anomaly detection," *IEEE Transactions on Image Processing*, vol. 34, pp. 5115–5130, 2025.

[6] J. Li, J. Li, C. Wang, F. J. Verbeek, T. Schultz, and H. Liu, "Outlier detection using iterative adaptive mini-minimum spanning tree generation with applications on medical data," *Frontiers in Physiology*, vol. Volume 14 - 2023, 2023. [Online]. Available: https://www.frontiersin.org/journals/physiology/articles/10.3389/fphys.2023.1233341

[7] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for iot malware detection with convolution neural network model," *Ieee Access*, vol. 8, pp. 96 899–96 911, 2020.

[8] D. Liu, Z. Cao, H. Jiang, S. Zhou, Z. Xiao, and F. Zeng, "Concurrent low-power listening: A new design paradigm for duty-cycling communication," *ACM Trans. Sen. Netw.*, vol. 19, no. 1, Dec. 2022. [Online]. Available: https://doi.org/10.1145/3517013

[9] G. Xu, L. Wang, S. Chen, L. Zhu, M. Guizani, and L. Shi, "Mpaee: A multipath adaptive energy-efficient routing scheme for low earth orbit-based industrial internet of things," *IEEE Internet of Things Journal*, vol. 12, no. 17, pp. 34 793–34 805, 2025.

[10] S. Liu, Y. Shen, J. Yuan, C. Wu, and R. Yin, "Storage-aware joint user scheduling and bandwidth allocation for federated edge learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 11, no. 1, pp. 581–593, 2025.

[11] X. Dai, Z. Xiao, H. Jiang, H. Chen, G. Min, S. Dustdar, and J. Cao, "A learning-based approach for vehicle-to-vehicle computation offloading," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 7244–7258, 2023.

[12] G. Xu, X. Fan, S. Xu, Y. Cao, X.-B. Chen, T. Shang, and S. Yu, "Anonymity-enhanced sequential multi-signer ring signature for secure medical data sharing in iomt," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 5647–5662, 2025.

[13] S. Srivastava, K. Kansal, S. Sai, and V. Chamola, "Secure cognitive health monitoring using a directed acyclic graph-based and ai-enhanced iomt framework," *Digital Communications and Networks*, vol. 11, no. 3, pp. 594–602, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864824001068

[14] S. Sai, K. S. Bhandari, A. Nawal, V. Chamola, and B. Sikdar, "An iomt-based incremental learning framework with a novel feature selection algorithm for intelligent diagnosis in smart healthcare," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 370–383, 2024.

[15] R.-H. Hsu, Y.-C. Wang, C.-I. Fan, B. Sun, T. Ban, T. Takahashi, T.-W. Wu, and S.-W. Kao, "A privacy-preserving federated learning system for android malware detection based on edge computing," in *2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*. IEEE, pp. 128–136, 2020.

[16] M. Abdel-Basset, H. Hawash, K. M. Sallam, I. Elgendi, K. Munasinghe, and A. Jamalipour, "Efficient and lightweight convolutional networks for iot malware detection: A federated learning approach," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 7164–7173, 2022.

[17] Y. Chen, S. He, B. Wang, Z. Feng, G. Zhu, and Z. Tian, "A verifiable privacy-preserving federated learning framework against collusion attacks," *IEEE Transactions on Mobile Computing*, vol. PP, pp. 1–17, 01 2024.

[18] P. Verma, N. Bharot, J. G. Breslin, D. O'Shea, A. K. Mishra, A. Vidyarthi, and D. Gupta, "Leveraging transfer learning domain adaptation model with federated learning to revolutionise healthcare," *Expert Systems*, vol. 42, no. 2, p. e13827, 2025.

[19] M. Asam, S. H. Khan, A. Akbar, S. Bibi, T. Jamal, A. Khan, U. Ghafoor, and M. R. Bhutta, "Iot malware detection architecture using a novel channel boosted and squeezed cnn," *Scientific Reports*, vol. 12, no. 1, p. 15498, 2022.

[20] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, vol. 204, p. 108693, 2022.

[21] Z. Li, C. Yan, X. Zhang, G. Gharibi, Z. Yin, X. Jiang, and B. A. Malin, "Split learning for distributed collaborative training of deep learning models in health informatics," in *AMIA Annual Symposium Proceedings*, vol. 2023, p. 1047, 2024.

[22] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," *arXiv:1909.09145*, 2019.

[23] S. Otoum, N. Guizani, and H. Mouftah, "On the feasibility of split learning, transfer learning and federated learning for preserving security in its systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 7462–7470, 2022.

[24] E. Samikwa, A. Di Maio, and T. Braun, "Ares: Adaptive resource-aware split learning for internet of things," *Computer Networks*, vol. 218, p. 109380, 2022.

[25] S. M. Popescu, S. Mansoor, O. A. Wani, S. S. Kumar, V. Sharma, A. Sharma, V. M. Arya, M. Kirkham, D. Hou, N. Bolan *et al.*, "Artificial intelligence and iot driven technologies for environmental pollution monitoring and management," *Frontiers in Environmental Science*, vol. 12, p. 1336088, 2024.

[26] R. S. Molina, V. Ninkovic, D. Vukobratovic, M. L. Crespo, and M. Zennaro, "Efficient split learning lstm models for fpga-based edge iot devices," *arXiv:2502.08692*, 2025.

[27] B.-H. Kim, A. Haldorai, and S. Suprakash, "A battery lifetime monitoring and estimation using split learning algorithm in smart mobile consumer electronics," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 3, pp. 5942–5951, 2024.

[28] S. Razdan and S. Sharma, "Internet of medical things (iomt): Overview, emerging technologies, and case studies," *IETE technical review*, vol. 39, no. 4, pp. 775–788, 2022.

[29] H. Babbar, S. Rani, and W. Boulila, "Ngmd: next generation malware detection in federated server with deep neural network model for autonomous networks," *Scientific Reports*, vol. 14, no. 1, p. 10898, 2024.

[30] K. Cao, H. Ding, B. Wang, L. Lv, J. Tian, Q. Wei, and F. Gong, "Enhancing physical-layer security for iot with nonorthogonal multiple access assisted semi-grant-free transmission," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 24 669–24 681, 2022.

[31] W. Zhang, S. Wang, P. Wang, Q. Fu, and X. Li, "Design of accelerometers-based puf for internet of things security," *IEEE Sensors Journal*, vol. 25, no. 17, pp. 32 984–32 992, 2025.

[32] M. Asiri, M. A. Khemakhem, R. M. Alhebshi, B. S. Alsulami, and F. E. Eassa, "Rpfl: A reliable and privacy-preserving framework for federated learning-based iot malware detection," *Electronics*, vol. 14, no. 6, p. 1089, 2025.

[33] S. Sai, V. Chamola, K.-K. R. Choo, B. Sikdar, and J. J. P. C. Rodrigues, "Confluence of blockchain and artificial intelligence technologies for secure and scalable healthcare solutions: A review," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5873–5897, 2023.

[34] L. Yu, Y. Li, S. Weng, H. Tian, and J. Liu, "Adaptive multi-teacher softened relational knowledge distillation framework for payload mismatch in image steganalysis," *Journal of Visual Communication and Image Representation*, vol. 95, p. 103900, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320323001505

[35] Y. Zhang, Z. Wang, M. Huang, M. Li, J. Zhang, S. Wang, J. Zhang, and H. Zhang, "S2dbft: Spectral–spatial dual-branch fusion transformer for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 63, pp. 1–17, 2025.

[36] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv:1812.00564*, 2018.

[37] S. Sai and V. Chamola, "A blockchain-enabled split learning framework with a novel client selection method for collaborative learning in smart healthcare," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 3, pp. 5887–5894, 2024.

[38] S. Sai, V. Hassija, V. Chamola, and M. Guizani, "Federated learning and nft-based privacy-preserving medical-data-sharing scheme for intelligent diagnosis in smart healthcare," *IEEE Internet of Things Journal*, vol. 11, no. 4, pp. 5568–5577, 2024.

[39] X. Fang, X. Chen, H. Chen, Z. Li, X. Chen, H. Huang, and W. Yao, "Joint optimization of sensing-communication-computing resource allocation for transparent and secure control in distribution

networks," *IEEE Transactions on Consumer Electronics*, vol. 71, no. 2, pp. 2974–2987, 2025.

[40] B. Cao, Y. Gu, Z. Lv, S. Yang, J. Zhao, and Y. Li, "Rfid reader anticollision based on distributed parallel particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3099–3107, 2021.

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

[42] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[43] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv:2003.00295*, 2020.