P3SL: Personalized Privacy-Preserving Split Learning on Heterogeneous Edge Devices

Wei Fan*, JinYi Yoon*, Xiaochang Li†, Huajie Shao†, Bo Ji*
*Department of Computer Science, Virginia Tech, Blacksburg, USA
†Department of Computer Science, William & Mary, Williamsburg, USA
*{fanwei, jinyiyoon, boji}@vt.edu, †{xli59, hshao}@wm.edu

***Abstract*—Split Learning (SL) is an emerging privacy-preserving technique that enables resource-constrained edge devices to participate in model training by partitioning models into client-side and server-side sub-models. While SL can reduce computational load on edge devices, it encounters significant challenges in heterogeneous environments with varying computing resources, communication capabilities, environmental conditions, and privacy requirements. Recent heterogeneous SL frameworks optimize split points for devices with varying resource constraints but often overlook personalized privacy requirements and local model customization under diverse environmental conditions. To address this, we propose P3SL, a Personalized Privacy-Preserving Split Learning framework for heterogeneous, resource-constrained edge systems. First, we design a personalized sequential split learning pipeline that allows each client to choose a customized split point and maintain local models tailored to its computational resources, environmental conditions, and privacy needs. Second, we formulate a bi-level optimization problem that empowers clients to determine their optimal split points without revealing private information (e.g., computational resources, environmental conditions, or privacy requirements) to the server. This approach balances energy consumption and privacy leakage while maintaining high accuracy. We have evaluated P3SL on seven edge devices—including four Jetson Nano P3450, two Raspberry Pi, and one laptop—using diverse models and datasets under varying environmental conditions. Results show that P3SL significantly reduces privacy leakage, lowers energy consumption by up to 59.12%, and consistently maintains high accuracy compared to state-of-the-art heterogeneous SL systems.**

***Index Terms*—Split Learning, Edge Computing, Heterogeneity, Personalized Privacy Protection**

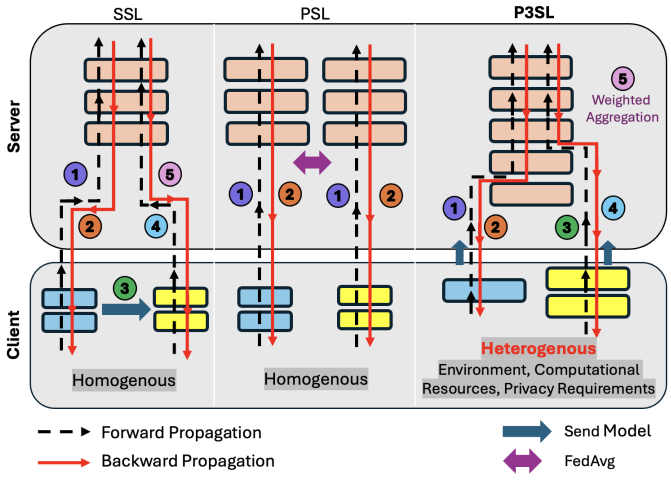


Fig. 1: Comparison of SSL, PSL, and P3SL frameworks: SSL (left): Sequential training with homogeneous split points, requiring inter-client model sharing and does not support personalization. PSL (center): Parallel training but still uses homogeneous split points, resulting in high server resource cost and reduced accuracy due to lack of personalization. **Our P3SL (right):** Personalized split points, adapting to heterogeneous conditions including computational resources, privacy requirements, and environmental settings.

I. Introduction

The rise of Internet-of-Things (IoT) devices has integrated them into daily life for tasks like sensing, monitoring, machine learning (ML), and intelligent decision-making [1], [2]. To protect data privacy, some research proposes training full ML models locally [3]. However, this approach is impractical for resource-constrained edge devices due to high energy consumption and prolonged training time. Split Learning (SL) addresses these challenges by partitioning ML models into client-side and server-side sub-models [4]. Unlike Federated Learning (FL) [5], which keeps the full model on clients, SL reduces client-side computational load by processing only a portion of the model.

In real-world SL applications, devices often operate with diverse computational resources and privacy requirements in heterogeneous environments. However, most existing SL approaches assume a homogeneous client environment [6]–[8], applying the same split point to all devices. The split point in SL defines where the neural network is divided, with clients handling layers before it and the server processing the rest. Sequential Split Learning (SSL) [4] and Parallel Split Learning (PSL) [9] are the two main homogeneous SL types, shown in Fig. 1. SSL trains clients sequentially with high accuracy and low server cost, but requires inter-client model sharing, limiting personalization and privacy. PSL allows concurrent training without model sharing but lowers accuracy and demands higher server resources, making it undesired for large-scale deployments. These approaches also overlook the varying capabilities and privacy needs in heterogeneous IoT ecosystems. For example, a tablet in a public area or a Home Assistant in a private space may jointly train a healthcare model [10]. The tablet, with more computational resources, may process less sensitive data, while the resource-limited

TABLE I: Comparison of P3SL with existing SL methods

Methods	Energy	Privacy	Personalized Model	Environments
SSL [4]	✗	✗	✗	✗
ARES [11]	✓	✗	✗	✗
ASL [12]	✓	✗	✗	✗
EdgeSplit [13]	✗	✗	✗	✗
P3SL (ours)	✓	✓	✓	✓

Home Assistant device handles highly sensitive private data. A uniform SL strategy would result in suboptimal performance, failing to meet the unique requirements of energy and privacy.

Existing works, such as ARES [11], ASL [12], and EdgeSplit [13], have studied heterogeneous SL frameworks that optimize split points based on individual resource constraints. These frameworks focus on factors like training latency and computational resources for each client, offering an improvement over uniform SL strategies. However, these approaches often neglect several critical issues: (i) *Privacy leakage across varying split points*: Different split points expose intermediate representations to varying levels of vulnerability [14], heightening the risk of data reconstruction attacks; (ii) *Energy consumption and power constraints under heterogeneous environments*: Real-world deployments often occur in diverse environmental conditions, such as varying temperature and humidity, which can significantly impact total energy consumption and peak instantaneous power usage; (iii) *Personalized client models and personalized privacy requirements*: In practice, clients have varying privacy requirements based on their computing resources and environments. A comparison of P3SL with state-of-the-art frameworks is shown in Table I.

To tackle the problems, we propose P3SL (**P**ersonalized **P**rivacy-**P**reserving **S**plit **L**earning), a novel SL framework that enables client devices to select optimal *personalized* split points tailored to varying privacy and energy needs in heterogeneous environmental conditions. P3SL addresses two key challenges: (i) how to effectively support numerous heterogeneous edge devices in SL, enabling private models with personalized split points while maintaining high accuracy; and (ii) how to determine optimal split point and privacy protection level for each device without revealing sensitive information (e.g., environments, computational resources, privacy requirements) to the server while ensuring high accuracy.

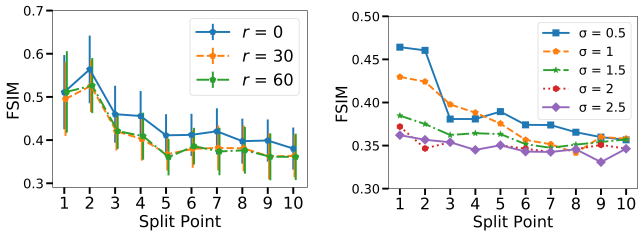
To the best of our knowledge, P3SL is the first work towards personalized privacy-preserving SL for heterogeneous edge devices operating under varying environmental conditions. We summarize the main contributions as follows:

- We propose P3SL, a framework featuring a personalized sequential SL pipeline that enables clients to train personalized models without inter-client model sharing, while maintaining high accuracy in heterogeneous, resource-constrained edge computing systems. This approach reduces update costs and enhances privacy protection against data reconstruction attacks, directly addressing the first challenge. We also design a weighted aggregation method that reduces the frequency of server communication, significantly improving communication efficiency.
- We formulate a bi-level optimization problem to jointly select split points and privacy protection levels for each device. By employing a meta-heuristic approach [15], [16], clients select their own split points and privacy levels based on privacy preferences without revealing sensitive information to the server. This enables the system to achieve an effective trade-off between energy consumption, privacy risk, and accuracy.
- We implement P3SL on a testbed of seven edge devices and evaluate its effectiveness using three model architectures across three datasets under varying environmental conditions. Our experimental results show that P3SL improves personalized privacy protection in terms of the Feature SIMilarity (FSIM) score [17] and significantly reduces energy consumption by up to 59.12% compared to existing SL systems while retaining a high accuracy.

II. Related Work

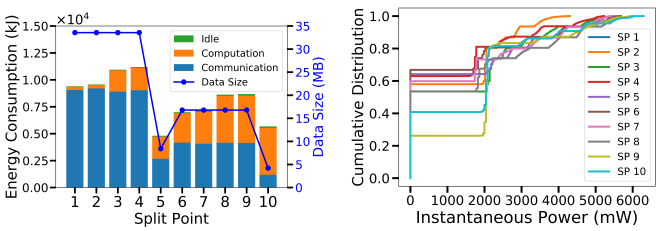
Split Learning Approaches. Existing SL frameworks, such as SSL [4], [18] and PSL [9], assume uniform split points across clients, simplifying implementation but failing to address the diverse resource constraints of heterogeneous environments. Recent heterogeneous SL works, such as ARES [11], ASL [12], and EdgeSplit [13], optimize split points based on computational latency, communication overhead, power consumption, accuracy requirements, and transmitted intermediate output size. However, they lack personalized privacy protection, which is essential in heterogeneous settings where privacy risks differ among clients. Additionally, EdgeSplit and ASL are evaluated only in simulations. Although ARES is implemented on real edge devices, it relies on a high-capacity server to process server-side model layers (i.e., the portions of the model executed on the server after the split point) for all clients concurrently, limiting its scalability in resource-constrained systems.

Defense Mechanisms against Data Reconstruction Attacks in SL. In split learning, attack methods such as input data reconstruction [14], [19] can recover raw data from transmitted model parameters or intermediate representations, leading to privacy leakage problems. Several defense mechanisms have been proposed to mitigate these risks, including adding noise to raw data [20], intermediate representation [21], [22], or model parameters [23], [24]. Although these approaches can effectively mitigate data reconstruction attacks, the performance depends on factors such as the layer from which outputs are generated (e.g., outputs from deeper layers are harder to recover [14]). Also, given the resource constraints, it is infeasible to fully commit to privacy protection without considering affordable computations in SL. Existing methods such as ASL and SSL do not provide personalized privacy protection for clients with heterogeneous resources. To address these challenges, P3SL offers a novel SL approach by integrating personalized privacy protection for heterogeneous resource-constrained edge devices.



(a) Privacy leakage at three different training epochs r (b) Defense with different variance σ values for Laplacian noise, $\mathcal{N}(0, \sigma^2)$

Fig. 2: Impact of split points on privacy leakage and defense with different variance



(a) Energy consumption and data size (b) CDF of instantaneous power

Fig. 3: Impact of different split points (SP) on energy consumption and instantaneous power

III. Motivation and Key Insights

In this section, we present case studies exploring the correlation between split points, energy consumption, and privacy leakage levels, motivating the design of P3SL.

A. Key Insights

We first evaluated the impact of personalized split points on privacy leakage, energy consumption, and power constraints per device. We implemented the data reconstruction attack of *UnSplit* [14] on the VGG16-BN [25] model using the CIFAR-10 [26] dataset. Privacy leakage is quantified using the FSIM [17] score, which measures similarity between original and reconstructed images. FSIM score ranges from 0 to 1, with higher values indicating a higher risk of privacy leakage.

Privacy Leakage Across Split Points. We explore privacy leakage across split points by varying the number of training epochs of 0, 30, and 60 epochs (where the total 100 epochs are required until saturation). We examine split points from Layer 1 to Layer 10, where higher indices mean that more layers are processed on the client before sending intermediate outputs to the server. As depicted in Fig. 2(a), deeper split points lead to lower FSIM scores, indicating better privacy protection as reconstruction input data becomes more difficult. Moreover, FSIM scores remain consistent across the three attack stages, since the *UnSplit* attack recovers input data and inverts the client model simultaneously. Thus, the data reconstruction results are not correlated with the stage of training.

Observation 1. *As evidenced by the FSIM scores, different split points present varying levels of privacy leakage; a shallower split point poses a higher risk of data reconstruction, necessitating aggressive noise addition.*

Defense Effectiveness. To mitigate data reconstruction attacks, we employed the noise addition approach from *NoPeekNN* [22], injecting Laplacian noise with zero mean and variance σ^2 into the intermediate outputs. We analyzed FSIM scores across noise levels σ ranging from 0.5 to 2.5 (incremented by 0.5). As shown in Fig. 2(b), we demonstrate that higher noise levels yield lower FSIM scores, indicating stronger privacy protection. However, excessive noise (e.g., σ = 2.0 or 2.5) diminishes distinctions among split points, leading to similar FSIM scores across all. This highlights a trade-off: shallower layers leak more privacy (as in Observation 1) and require stronger noise, potentially harming accuracy.

Observation 2. *To achieve a desirable privacy protection for each split point, it is essential to determine the minimum noise injection to reach the target privacy threshold.*

Energy Consumption and Power Constraints. We evaluate the energy and power implications of split points in an SSL setting using four 4 GB NVIDIA Jetson Nano P3450 devices [27]. Fig. 3(a) shows the size of intermediate representations at different split layers (blue trend) and the average energy consumption across communication, computation, and idle states (bar plot). Results indicate that communication energy scales proportionally with the size of intermediate representations. Deeper split points produce smaller representations (i.e., lower data dimensions), reducing communication energy but increasing computational energy due to more client-side computation. Layers with the same intermediate size ideally yield the same communication energy, but deeper layers still incur higher computational energy. Thus, total energy consumption rises with deeper split points, making it essential to choose a split point with lower energy consumption.

Furthermore, edge devices often face diverse environmental conditions, such as temperature variations, seasonal changes, or location-specific factors, which can cause overheating issues even in identical devices. Fig. 3(b) presents the cumulative distribution of instantaneous power measured by the Jetson Nano devices, illustrating that deeper split points result in higher peak instantaneous power. To prevent overheating, it is crucial to ensure instantaneous power stays within the peak limit set for each client based on their environment conditions.

Observation 3. *Deeper split points reduce communication energy but increase computational energy consumption, indicating that optimizing split points involves balancing these trade-offs to align with the power constraints of edge devices.*

These observations suggest a trade-off between privacy protection and energy/power consumption across split points: shallower split points increase (i) privacy leakage; (ii) communication energy; while reducing (iii) computational energy.

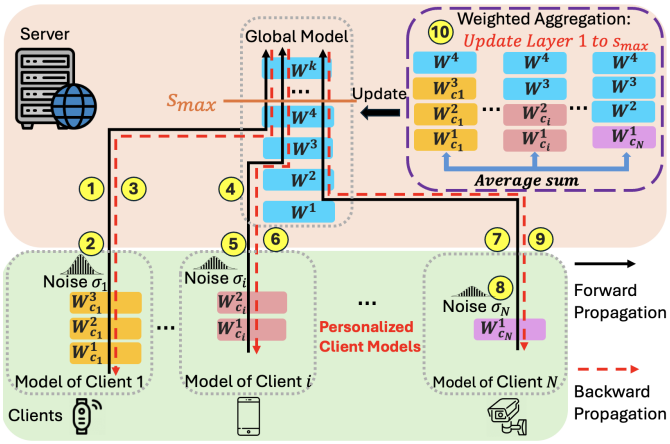


Fig. 4: System architecture of P3SL: with $\boldsymbol{s}$ of split points and $\boldsymbol{\sigma}$ of privacy protection (noise) levels, clients generate intermediate representations, inject noise, and upload them to the server for ①-⑨ sequential model training. Every R epoch, clients upload their local models to the server for ⑩ weighted aggregation. The process ①-⑩ repeats until the global model converges.

B. Design Goals

Our design aims to achieve two important goals:

(i) **Support Personalized Models.** To design a training pipeline that enables each client to maintain a personalized model, split point, and privacy protection, accommodating resource-heterogeneous devices in SL settings.

(ii) **Balance Energy Consumption and Privacy Leakage Risk.** Allow clients choose personalized split points with privacy protection levels to balance energy consumption and privacy leakage while ensuring high global model accuracy. Conduct an approach that effectively addresses these trade-offs to optimize overall performance across heterogeneous environments.

IV. P3SL Design

We design P3SL with an SSL system (Fig. 4) that supports edge devices to have heterogeneous split points and personalized privacy protection in two ways: 1) personalized model to keep the private model locally; 2) personalized privacy protection to adjust the injected noise level. Clients can choose different split points and noise levels for their intermediate representations before uploading to the server. The global model is trained sequentially without inter-client model sharing. Our system model is detailed below.

A. P3SL Architecture

Let $\mathcal{C} := \{1, 2, \cdots, N\}$ denote a set of N heterogeneous devices with varying computational capabilities. Each client $i \in \mathcal{C}$ has its local model $\boldsymbol{W}_{c_i}$ and a local dataset $\mathcal{D}_i$ with $|\mathcal{D}_i|$ labeled samples. The local dataset is defined as $\mathcal{D}_i := \{(x_{i,j}, y_{i,j})\}_{j=1}^{|\mathcal{D}_i|}$, where $x_{i,j}$ is the j-th input data sample, and $y_{i,j}$ is its corresponding label. Hence, the global dataset $\mathcal{D}$ is denoted as $\mathcal{D} = \bigcup_{i \in \mathcal{C}} \mathcal{D}_i$.

For simplicity, we define $\boldsymbol{W}^{a:b} := [\boldsymbol{W}^a, \boldsymbol{W}^{a+1}, \cdots, \boldsymbol{W}^b]$ as the model containing layers from a to b. In P3SL, all clients collaborate to train the global model $\boldsymbol{W} := \boldsymbol{W}^{1:k}$, where k is the total number of layers in the global model. The global model $\boldsymbol{W}$ serves as the shared server model and is partitioned at a split point s_i into two sub-models. For client i, the first s_i layers are located on the client side, denoted as $\boldsymbol{W}_{c_i} := \boldsymbol{W}_{c_i}^{1:s_i}$. The remaining layers $s_i + 1$ to k are on server side, denoted as $\boldsymbol{W}_{g_i} := \boldsymbol{W}_{g_i}^{s_i+1:k}$. The server also decides the maximum allowable split point $s_{\max}$ for all clients, and each client i selects $s_i \in \{1, 2, \ldots, s_{\max}\}$ based on local privacy requirements and energy consumption constraints. By definition, $1 \le s_i \le s_{\max} \le k$.

The overall training procedure of personalized sequential split learning in P3SL follows the steps illustrated in Fig. 4:

(i) *Forward Propagation on Client-Side Model:* After initializing the global model at the server, P3SL sequentially selects each client for training. The selected i-th client samples local labeled data $(x_{i,j}, y_{i,j}) \in \mathcal{D}_i$ and performs forward propagation through its local model to generate intermediate representations $\hat{z}_{i,j} := g(x_{i,j} \mid \boldsymbol{W}_{c_i})$. Here, $g(x \mid \boldsymbol{W})$ maps input data x to $\hat{z}$ with model $\boldsymbol{W}$ (see ①④⑦).

(ii) *Privacy-Preserving Intermediate Representation Transmission:* The client i injects Laplacian noise $\eta_{i,j} \sim \mathcal{N}(0, {\sigma_i}^2)$ into $\hat{z}_{i,j}$, where σ_i is the personalized privacy protection level. Then, the noise-injected representation $\hat{z}_{i,j} + \eta_{i,j}$ and the corresponding label $y_{i,j}$ are sent to the server (see ②⑤⑧).

(iii) *Forward Propagation on Server-Side Model:* The server receives the privacy-protected intermediate representation, processes it through the remaining layers of the global model $\boldsymbol{W}_{g_i} := \boldsymbol{W}^{s_i+1:k}$, and generate the final prediction $\hat{y}_{i,j}$.

(iv) *Gradient Calculation and Backward propagation:* The server computes gradients based on the loss $\mathcal{L}(\hat{y}_{i,j}, y_{i,j} \mid \boldsymbol{W}_{c_i} \oplus \boldsymbol{W}_{g_i})$, where $\oplus$ denotes model concatenation: $\boldsymbol{W}^{a:b} \oplus \boldsymbol{W}^{c:d} := [\boldsymbol{W}^a, \boldsymbol{W}^{a+1}, \cdots, \boldsymbol{W}^b, \boldsymbol{W}^c, \boldsymbol{W}^{c+1}, \cdots \boldsymbol{W}^d]$. During backpropagation, the server only updates the layer in $\boldsymbol{W}_{g_i}$ and sends the gradients back to the client. The client then completes the backward propagation and updates its model parameters (see ③⑥⑨).

(v) *Model Aggregation and the Global Model Updates:* After R epochs, clients upload their local parameters to update the global model's first $s_{\max}$ layers $\boldsymbol{W}^{1:s_{\max}}$. All uploaded parameters $\boldsymbol{W}_{c_i}$ will be aggregated using Eq. (1). Missing layers from $s_i + 1$ to $s_{\max}$ are filled with corresponding global layers. The global model $\boldsymbol{W}$ is updated as:

$$\boldsymbol{W} = \left(\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{W}_{c_i} \oplus \boldsymbol{W}^{s_i+1:s_{\max}})\right) \oplus \boldsymbol{W}^{s_{\max}+1:k}. \quad (1)$$

This aggregation ensures $\boldsymbol{W}^{1:s_{\max}}$ is not overly biased by clients with larger s_i, and the aggregated model is not distributed to preserve personalization of client models (see ⑩).

This process repeats until all clients finish updates and the system loss is minimized.

SP \ σ	0.00	0.05	…	2.50
1	0.53	0.53		0.36
2	0.53	0.52		0.35
…				
$s_{\max}$	0.37	0.35		0.33

(a) *Privacy Leakage Table* profiled by server

S_i	$E_i^{\text{total}}(S_i)$	$p_i^{\text{peak}}(S_i)$
1	903kJ	4311mW
2	951kJ	4331mW
…		
$S_{\max}$	633kJ	6297mW

(b) *Energy and Power Consumption Table* profiled by each client i

Fig. 5: An example of profiling tables

B. Profiling

We introduce the profiling process, including *Privacy Leakage Table* (server-side) and *Energy and Power Consumption Table* (client-side), essential for solving the bi-level optimization problem (Section V). The server also sets a minimum accuracy threshold A_{min} to ensure global model accuracy.

Constructing *Privacy Leakage Table*. Assume that the attacker can access the model architecture and intermediate representations. Since reconstruction attack performances are similar in all training stages (shown in Fig. 2(a)), we apply attacking before training [14]. Privacy leakage is measured by FSIM [17], which quantifies similarity between original and reconstructed data. Since clients jointly train the global model, the server simulates attacks on a public dataset to generate the *Privacy Leakage Table*. As shown in Fig. 5(a), it records FSIM scores for split points 1 to $s_{\max}$ across noise levels from 0.00 to 2.50 (0.05 interval) and is then distributed to clients.

Constructing *Energy and Power Consumption Table*. Each client profiles energy consumption and peak instantaneous power for split points $s_i \in [1, s_{\max}]$. For each s_i, the i-th client generates an *Energy and Power Consumption Table* (Fig. 5(b)) containing: (i) average total energy $E_i^{\text{total}}(s_i)$; and (ii) peak instantaneous power $p_i^{\text{peak}}(s_i)$. Meanwhile, the client tests out its (iii) maximum power threshold $P_i^{\max}$ to avoid overheating.

Minimum Accuracy Threshold Determination. When initiating P3SL training, the server sets the minimum accuracy threshold A_{min}, which is the global accuracy required to complete optimization. To establish it, the server simulates training without noise injection on a public dataset that closely matches the data distribution of clients, yielding a reference accuracy A_{ref} as the ideal baseline. During this phase, the server also tunes and shares hyperparameters with clients. The threshold A_{min} is then computed as:

$$A_{\text{min}} = \beta \cdot A_{\text{ref}}, \tag{2}$$

where β is a server-defined discount factor representing acceptable accuracy loss for privacy, fixed throughout training.

V. Problem Formulation and Proposed Solution

In real-world scenarios, clients are often reluctant to share sensitive information with the server due to privacy concerns. If the server solely determines split points and noise levels for all clients, it requires access to client-specific details (e.g., environment, computational resources, and privacy requirements). To address this, P3SL facilitates joint decision-making between server and clients to optimize system performance.

In this section, we formulate a bi-level optimization problem (Section V-A) and propose a solution (Section V-B) that allows the server and clients to determine the optimal noise levels and split points jointly. This solution effectively balances the trade-off between privacy protection and energy efficiency while maintaining high global accuracy.

A. Bi-Level Optimization Problem Formulation

We define two decision variables: the privacy protection (noise) level vector $\boldsymbol{\sigma} := [\sigma_1, \sigma_2, \cdots, \sigma_N]$, where σ_i is the noise level for client i, and the split point vector $\boldsymbol{s} := [s_1, s_2, \cdots, s_N]$, where s_i is client i's split point.

In the lower-level optimization, given the upper-level decision on the noise level σ_i from the server, the client i decides its optimal split point s_i by minimizing the local objective function f. Since minimizing total energy $E_i^{\text{total}}(s_i)$ and privacy leakage $\text{FSIM}(\sigma_i, s_i)$ may conflict, f is formulated as their weighted sum. The total energy $E_i^{\text{total}}(s_i)$ is taken from the *Energy and Power Consumption Table* (Fig. 5(b)), and the privacy leakage $\text{FSIM}(\sigma_i, s_i)$ is from the *Privacy Leakage Table* (Fig. 5(a)). The trade-off is governed by the privacy sensitivity coefficient $\alpha_i \in [0, 1]$, reflecting the privacy preference of the client i. A higher α_i favors more privacy protection (more noise and deeper split points), while a lower α_i favors energy efficiency (reduced energy with shallower split points). The local objective function f for each client i is formulated as:

$$f(\sigma_i, s_i) := \alpha_i \cdot \text{FSIM}(\sigma_i, s_i) + (1 - \alpha_i) \cdot E_i^{\text{total}}(s_i). \tag{3}$$

To optimize the privacy across all clients while maintaining high global model accuracy, the server (upper-level) decides the privacy protection level vector $\boldsymbol{\sigma}$ to minimize the total privacy leakage $\text{FSIM}(\sigma_i, s_i)$ across all clients. Meanwhile, the server ensures that the global model accuracy $G_{\text{acc}}(\boldsymbol{\sigma}, \boldsymbol{s})$ remains above the minimum accuracy threshold A_{min}. This bi-level optimization problem is formulated as:

$$\min_{\boldsymbol{\sigma}, \boldsymbol{s}} \; F(\boldsymbol{\sigma}, \boldsymbol{s}) := \sum_{i=1}^{N} \text{FSIM}(\sigma_i, s_i) \tag{4a}$$

$$\text{s.t. } s_i \in \arg\min_{s_i} f(\sigma_i, s_i), \forall i \in \mathcal{C}, \tag{4b}$$

$$G_{\text{acc}}(\boldsymbol{\sigma}, \boldsymbol{s}) \geq A_{\text{min}}, \tag{4c}$$

$$p_i^{\text{peak}}(s_i) \leq P_i^{\max}, \forall i \in \mathcal{C}. \tag{4d}$$

Here, Eq. (4b) states that each client i selects s_i by minimizing the local objective function f; Eq. (4c) ensures that the global accuracy $G_{\text{acc}}(\boldsymbol{\sigma}, \boldsymbol{s})$ stays above the threshold A_{min}; Eq. (4d) ensures client i's peak power $p_i^{\text{peak}}(s_i)$ does not exceed its maximum threshold $P_i^{\max}$, preventing overheating. The value of $P_i^{\max}$ is from client-side profiling process (Section IV-B).

B. Proposed Solution

We employ a meta-heuristic algorithm [15] to solve the bi-level optimization problem. The upper-level problem (Eq. (4a)) optimizes privacy across clients while ensuring global accuracy meets the threshold. The lower-level problem (Eq. (3)) balances energy consumption and privacy leakage for each client. These are solved sequentially to allow iterative refinement and convergence toward an optimal solution [28]. The process begins with the server generating an initial solution based on the upper-level objective and distributing it to clients. Each client then selects its split point by minimizing the local objective and sends its solution back to the server. This nested process repeats until the overall optimization is convergent.

We implement this approach in the P3SL framework through three key steps:

(i) Initial Upper-Level Solution. The server generates a *Noise Assignment Table* based on the *Privacy Leakage Table*, mapping noise levels to split points to ensure low privacy leakage. The table is then distributed to the clients, allowing them to select the optimal split points.

(ii) Initial Lower-Level Solution. Each client i first sets its personalized privacy sensitivity coefficient $\alpha_i \in [0, 1]$. Then, it identifies the deepest split point $s_{\text{max}}^{(c_i)}$ that avoids overheating, and the point with lowest energy consumption $s_{\text{min}}^{(c_i)}$. If energy increases with depth, the client selects the optimal split point from 1 to $s_{\text{max}}^{(c_i)}$; if it decreases, the split point choice is between $s_{\text{min}}^{(c_i)}$ and $s_{\text{max}}^{(c_i)}$. Finally, the client minimizes the local objective function (Eq. (3)) using the server-provided *Noise Assignment Table*, then sends its chosen optimal split point s_i to the server.

(iii) Iterative Optimizing. The server constructs $\boldsymbol{s}$ and $\boldsymbol{\sigma}$, performs sequential training, and computes global accuracy $G_{\text{acc}}(\boldsymbol{\sigma}, \boldsymbol{s})$. If accuracy meets or exceeds A_{min}, optimization is finished. Otherwise, the server updates the *Noise Assignment Table* using reassignment strategies and redistributes it to clients. Clients then re-optimize their split points as in (ii), repeating the process until global accuracy reaches A_{min}.

Initial Noise Assignment Table. To generate the initial *Noise Assignment Table*, the server defines an FSIM threshold T_{FSIM} to prevent accurate classification of the reconstructed data. Specifically, the server uses a well-trained model to evaluate the reconstructed data under varying noise levels. Along with the *Privacy Leakage Table*, T_{FSIM} is set where the classification accuracy drops below $\frac{1}{N_{\text{class}}}$, with N_{class} as the number of classes. The server then selects, for each split point, the minimum noise level that keeps FSIM below T_{FSIM}, forming the *Noise Assignment Table*, which is distributed to clients for their initial split point selection.

Noise Reassignment Strategies. If global accuracy $G_{\text{acc}}(\boldsymbol{\sigma}, \boldsymbol{s})$ falls below A_{min}, the server reduces noise levels to enhance model utility. The server updates the *Noise Assignment Table* and redistributes it to clients. Denote A^t as the global accuracy in round t, and σ_i^t as noise level for the client i in round t. For round $t + 1$, the server updates σ_i as:

$$\sigma_i^{t+1} = \sigma_i^t \times (1 - 2 \cdot (A_{\text{min}} - A^t)). \tag{5}$$

A larger difference between A^t and A_{min} results in greater a noise reduction, thereby improving the global accuracy.

VI. Evaluation

This section addresses key research questions (RQs) on P3SL's privacy protection, energy efficiency, adaptability, and overall performance through comprehensive evaluations.

- RQ1: How does P3SL compare to baseline methods in terms of privacy leakage, energy consumption, and model accuracy under heterogeneous environmental conditions?
- RQ2: How flexible is P3SL in adapting to varying environmental conditions, such as temperature and cooling settings, and maintaining consistent performance?
- RQ3: How effective is P3SL for personalized privacy protection for individual clients with diverse resource constraints and privacy requirements?

A. Implementation and Deployment

Device Settings. We used four 4GB NVIDIA Jetson Nano (P3450) devices, two 4GB Raspberry Pi devices, and one GPU-less laptop as edge devices, along with one central server with dual NVIDIA GeForce RTX 4090 GPUs. Network connections and data transmission are established via `WebSocket` APIs. Power consumption is monitored in real time using the Kasa system [29].

Environment Settings. We conduct experiments in two controlled environments. In each setting, devices are placed in separate rooms with varying temperatures, and their cooling fans are toggled accordingly. For the laptop (Client 7), the fan operates automatically. Detailed environmental conditions are listed in Table II. To address potential overheating, we measure the deepest (i.e., maximum) split points $s_{\text{max}}^{(c_i)}$ by monitoring each client's instantaneous power consumption.

Datasets. We evaluate the system using (i) CIFAR-10 [26], (ii) Fashion-MNIST [30], and (iii) Flower-102 [31] datasets.

Model Architecture, training setup and Personalized Settings. We leverage three models ranging from lightweight to large-scale: ResNet18 (11M), ResNet101 (43M), and VGG16-BN (135M), with the deepest split point $s_{\text{max}} = 10$. Each client trains with a batch size of 256 and a learning rate of 0.01; models are aggregated every 5 epochs. P3SL applies sleep-awake scheduling [32], recording energy only during *awake* mode to reduce idle consumption. Clients use personalized privacy sensitivity coefficients to reflect their energy–privacy trade-off preferences: $\alpha_1 = 0.4, \alpha_2 = 0.2, \alpha_3 = 0.5, \alpha_4 = 0.9, \alpha_5 = 0.7, \alpha_6 = 0.3, \alpha_7 = 0.8$. The discount factor is set to $\beta = 5\%$, representing the maximum global accuracy sacrifice tolerated for clients' enhanced privacy.

Evaluation Metrics. We evaluate system performance using three metrics: (i) overall and personalized privacy leakage, (ii) energy consumption, and (iii) accuracy. Overall privacy leakage, $\text{FSIM}_{\text{total}}$, reflects system-wide leakage and is computed by averaging the FSIM scores of all clients over five rounds. We also investigate each client's personalized leakage, denoted as FSIM. A lower FSIM indicates stronger privacy, with even a 0.025 drop marking a significant reduction [17]. Energy

TABLE II: Two environment condition settings, where AC Temp. refers to the set temperature of the air conditioner

Client ID	Environment Setting A		Environment Setting B	
	AC Temp.	Cooling Fan	AC Temp.	Cooling Fan
1 (Jetson Nano)	30 $^\circ C$	OFF	30 $^\circ C$	ON
2 (Jetson Nano)	30 $^\circ C$	ON	20 $^\circ C$	OFF
3 (Jetson Nano)	20 $^\circ C$	OFF	15 $^\circ C$	OFF
4 (Jetson Nano)	20 $^\circ C$	ON	15 $^\circ C$	ON
5 (Raspberry Pi)	20 $^\circ C$	OFF	20 $^\circ C$	OFF
6 (Raspberry Pi)	20 $^\circ C$	ON	20 $^\circ C$	ON
7 (Laptop)	20 $^\circ C$	AUTO	20 $^\circ C$	AUTO

consumption, $\overline{E}_{\text{total}}$, refers to the average per-epoch energy consumption of all edge devices, including communication, computation, and idle states. Lastly, accuracy represents the global model's testing accuracy, indicating its utility.

Baselines. We compare P3SL against three state-of-the-art baselines: (i) *ASL* [12] and (ii) *ARES* [11], both supporting heterogeneous split points in PSL; (iii) *Sequential Split Learning (SSL)* [4], a classic SL framework enforcing homogeneous split points. SSL requires transmitting model parameters between clients, thus limiting its support for heterogeneous split points. Since none of these baselines account for privacy leakage, we ensure a fair comparison by applying the same noise injection with the same variance $\mathcal{N}(0, \sigma^2 = 2.5^2)$ as used in P3SL to all baselines.

B. RQ1: Privacy Leakage, Energy Consumption, and Global Model Accuracy

We first evaluate the overall performance of P3SL in privacy leakage, energy consumption, and global model accuracy compared to the baselines under environmental setting A.

Privacy Leakage and Accuracy. As shown in Table III, P3SL consistently reduces overall privacy leakage while maintaining high global accuracy, leveraging sequential SL without overloading the server. Although all methods apply the same noise injection defense, P3SL outperforms baselines by jointly optimizing split points and privacy levels per client via bi-level optimization, ensuring a more appropriate trade-off between privacy protection and energy consumption. In contrast, the baselines do not account for privacy leakage when selecting split points, leading to inefficient protection. Some clients may be overprotected with large noise, especially when applied to deeper layers, which yields minimal privacy gains but significantly harms accuracy. P3SL also preserves personalized client models by avoiding parameter sharing, further enhancing privacy. By integrating privacy into personalized split point selection, P3SL enhances overall privacy protection without compromising much accuracy. While P3SL has a bit longer training time than PSL methods (e.g., 418s/epoch for ResNet18 with F-MNIST vs. 252s/epoch for ARES and ASL), it remains practical, and the improved accuracy from sequential SL justifies the additional cost.

Energy Consumption. As shown in Table III, P3SL reduces total energy consumption by up to 38.68% and 59.12% compared to ASL and SSL, respectively. ASL's high energy consumption results from its reliance on PSL, which requires devices to remain actively engaged in computation for extended periods while waiting for stragglers to finish training. ASL also lacks a parallel execution management, further leading to energy inefficiency. In contrast, SSL's energy inefficiencies primarily stem from frequent communication, as model parameters must be repeatedly transmitted to initialize adjacent clients' training. We address these issues by employing a weighted aggregation technique, reducing the frequency of parameter uploads to the server and eliminating the need to distribute aggregated weights back to clients. Additionally, the use of sleep-wake scheduling minimizes energy consumption during idle time in the sequential training process.

TABLE III: Accuracy, privacy leakage, and energy consumption performance compared to baselines: smaller $\text{FSIM}_{\text{total}}$ and $\overline{E}_{\text{total}}$ indicate better privacy protection and less energy consumption for environment setting A (See Table II)

Model	Dataset	System	Accuracy	$\text{FSIM}_{\text{total}}$	$\overline{\text{E}}_{\text{total}}$ (kJ)
VGG16-BN	CIFAR-10	**P3SL**	**0.90** (↑)	**2.52** (↓)	**4390** (↓)
		ASL	0.84	2.55	6503
		ARES	0.85	2.57	6452
		SSL	0.86	2.63	8446
	Fashion-MNIST	**P3SL**	**0.92** (↑)	**2.52** (↓)	**5953** (↓)
		ASL	0.85	2.60	9443
		ARES	0.84	2.58	9393
		SSL	0.84	2.59	14562
	FLOWER-102	**P3SL**	**0.89** (↑)	**2.52** (↓)	**617** (↓)
		ASL	0.84	2.56	902
		ARES	0.83	2.54	894
		SSL	0.86	2.57	1335
ResNet18	CIFAR-10	**P3SL**	**0.91** (↑)	**2.51** (↓)	**6980** (↓)
		ASL	0.86	2.59	9896
		ARES	0.85	2.58	9994
		SSL	0.84	2.53	13582
	Fashion-MNIST	**P3SL**	**0.92** (↑)	**2.51** (↓)	**9566** (↓)
		ASL	0.86	2.59	13744
		ARES	0.88	2.63	13588
		SSL	0.85	2.58	19942
	FLOWER-102	**P3SL**	**0.91** (↑)	**2.51** (↓)	**1004** (↓)
		ASL	0.84	2.60	1482
		ARES	0.84	2.59	1421
		SSL	0.88	2.62	2005
ResNet101	CIFAR-10	**P3SL**	**0.92** (↑)	**2.50** (↓)	**7012** (↓)
		ASL	0.85	2.59	10799
		ARES	0.86	2.61	11023
		SSL	0.89	2.61	12811
	Fashion-MNIST	**P3SL**	**0.94** (↑)	**2.50** (↓)	**9467** (↓)
		ASL	0.87	2.61	15077
		ARES	0.87	2.60	15231
		SSL	0.91	2.62	16870
	FLOWER-102	**P3SL**	**0.91** (↑)	**2.45** (↓)	**1023** (↓)
		ASL	0.87	2.59	1600
		ARES	0.90	2.63	1507
		SSL	0.87	2.60	2346

Remark 1. *Personalized split points with personalized models in* P3SL *significantly reduce privacy leakage risks and system energy consumption while maintaining high accuracy.*

C. RQ2: Impact of Heterogeneous Environment Settings

To evaluate the impact of environmental settings, we compare total system privacy leakage ($\text{FSIM}_{\text{total}}$) and accuracy across P3SL, ASL, ARES, and SSL under two heterogeneous settings in Table IV. While environmental differences lead to

TABLE IV: Accuracy, privacy leakage, and energy consumption performance for VGG16-BN with CIFAR-10 dataset compared to baselines for heterogeneous environment settings

Environment	System	Accuracy	$\text{FSIM}_{\text{total}}$	$\overline{\text{E}}_{\text{total}}$ (kJ)
Environment A	**P3SL**	**0.90** (↑)	**2.52** (↓)	**4390** (↓)
	ASL	0.84	2.55	6503
	ARES	0.85	2.57	6452
	SSL	0.86	2.63	8446
Environment B	**P3SL**	**0.91** (↑)	**2.51** (↓)	**4433** (↓)
	ASL	0.86	2.56	6299
	ARES	0.87	2.57	6168
	SSL	0.89	2.60	8126

TABLE V: The effect of heterogeneous split points and personalized privacy protection using VGG16-BN in environment setting A. FSIM represents the privacy leakage level (higher is worse). Small FSIM differences indicate notable privacy variation. α_i is a personalized privacy sensitivity coefficient. The FSIM is shown both before and after noise injection.

Client ID	Split Point	α_i	Noise Injection	FSIM (Before → After)
1 (Jetson Nano)	2	0.4	$\mathcal{N}(0, 1.65^2)$	0.53 → **0.36**
2 (Jetson Nano)	3	0.2	$\mathcal{N}(0, 1.15^2)$	0.43 → **0.36**
3 (Jetson Nano)	5	0.5	$\mathcal{N}(0, 0.2^2)$	0.41 → **0.37**
4 (Jetson Nano)	10	0.9	$\mathcal{N}(0, 0.02^2)$	0.37 → **0.36**
5 (Raspberry Pi)	1	0.7	$\mathcal{N}(0, 2.15^2)$	0.53 → **0.37**
6 (Raspberry Pi)	2	0.3	$\mathcal{N}(0, 1.65^2)$	0.53 → **0.36**
7 (Laptop)	10	0.8	$\mathcal{N}(0, 0.02^2)$	0.37 → **0.35**

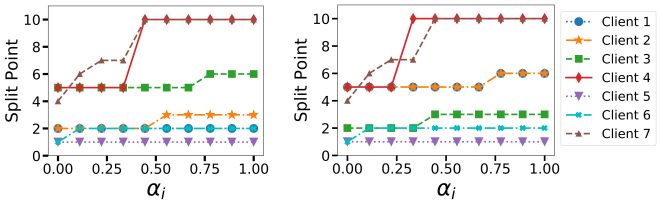
varied energy profiles and optimal split points per client, P3SL consistently achieves the highest accuracy and lowest privacy leakage. This demonstrates the effectiveness of bi-level optimization design in maintaining high accuracy and minimizing system privacy leakage across heterogeneous environments.

Remark 2. *P3SL demonstrates robustness in maintaining both high accuracy and low privacy leakage across varying environments through its adaptive noise adjustment and bi-level optimization mechanism.*

D. RQ3: Personalized Privacy Evaluation

We evaluate how P3SL enables personalized privacy protection for clients with heterogeneous resource constraints and privacy requirements. We further analyze the impact of varying the privacy sensitivity coefficient α_i.

Personalized Noise Injection and Privacy Requirements. As shown in Table V, each client in P3SL selects its split point based on computational resources, privacy preference, and power constraints. Personalized privacy protection in P3SL encompasses two components: personalized noise injection and the personalized privacy requirement, represented by the privacy sensitivity coefficient α_i. Clients with higher α_i prioritize privacy protection by selecting deeper split points, which inherently offer stronger privacy due to more number of layers involved. For instance, Client 4 and Client 7, with high α_i, select split point 10, requiring minimal noise injection ($\sigma = 0.02$) to achieve privacy preservation. Their corresponding FSIM reduces from $0.366 \rightarrow 0.355$ and $0.366 \rightarrow 0.354$, respectively. In contrast, clients with lower α_i prioritize energy efficiency, selecting shallower split points where privacy leakage risks are higher. To compensate, these clients require higher levels of noise injection to protect privacy. For example, Client 2 and Client 6, with low α_i, select split point 3 ($\sigma = 1.15$) and split point 2 ($\sigma = 1.65$), reducing FSIM from $0.428 \rightarrow 0.356$ and $0.532 \rightarrow 0.360$, respectively. This approach reflects personalized privacy protection, as clients adapt their split point selection and noise injection based on their α_i, significantly reducing FSIM across all clients.



(a) Environment setting A (b) Environment setting B

Fig. 6: Optimal split point selection for different personalized privacy sensitivity coefficient α_i

Impact of Personalized Privacy Sensitivity Coefficient α_i. We analyze the impact of varying α_i from $0 \rightarrow 1$ in 0.1 intervals for each client. As shown in Fig. 6 in two environment settings (Table II), clients with smaller α_i prioritize energy efficiency and prefer shallower split points. Conversely, as α_i increases, clients prefer deeper split points to enhance privacy protection. Additionally, some split points in the middle appear to be sweet spots, offering smaller intermediate representations that save communication energy while providing better privacy protection compared to shallower split points.

Remark 3. P3SL *suggests that personalized privacy preservation can significantly mitigate privacy risks tailored to each heterogeneous resource-constrained client and environment.*

VII. Discussion

This section discusses additional privacy concerns and the server profiling cost in P3SL.

Additional Privacy Concerns. Although P3SL enhances privacy through noise-injected intermediate outputs, risks remain during model and label sharing. Gradient Leakage Attacks (GLA) can exploit gradients from client-server communication to reconstruct inputs [33]; however, our use of noise significantly limits such risks [34]. Label leakage, while not affecting our threat model, can be mitigated by integrating techniques like U-shaped SL [35], allowing clients to retain labels locally for further privacy improvement.

Server Profiling Cost. P3SL requires one-time server-side profiling to construct the *Privacy Leakage Table*, introducing slight computational overhead. As privacy leakage is model-dependent, a single dataset suffices for this step. Centralized profiling ensures consistent privacy normalization across clients and avoids additional computation on resource-constrained devices, supporting fair and consistent privacy protection for all clients while optimizing resource efficiency.

VIII. Conclusion

We proposed P3SL, a novel SL framework that enables clients to maintain personalized privacy protection tailored to their heterogeneous resource constraints and deployment environments. P3SL incorporates a sequential training architecture with a weighted aggregation technique, allowing each edge device to maintain personalized local models, privacy protection, and heterogeneous split points. Additionally, P3SL employs bi-level optimization to let each client strategically select optimal split points, effectively balancing energy efficiency, privacy leakage risk, and high accuracy. We implemented and deployed P3SL on various edge devices and conducted extensive evaluations. Results consistently demonstrated that P3SL can effectively achieve better personalized privacy protection and less energy consumption while maintaining high accuracy, validating its effectiveness in heterogeneous, resource-constrained distributed learning environments.

For future work, we will extend P3SL to include models beyond image classification (e.g., transformer architectures) and other data types (e.g., tabular and time-series data). Additionally, incorporating advanced client selection mechanisms that prioritize high-quality data offers further potential to enhance global model robustness and training performance.

Acknowledgment

This research was supported in part by NSF grant CNS-2315851, the Commonwealth Cyber Initiative (CCI), and a Virginia Tech Presidential Postdoctoral Fellowship.

References

[1] S. Ye, L. Zeng, X. Chu, G. Xing, and X. Chen, "Asteroid: Resource-efficient hybrid pipeline parallelism for collaborative dnn training on heterogeneous edge devices," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 312–326.

[2] Thapa *et al.*, "Splitfed: When federated learning meets split learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8485–8493.

[3] S. Zhu, T. Voigt, J. Ko, and F. Rahimian, "On-device training: A first overview on existing systems," *arXiv preprint arXiv:2212.00824*, 2022.

[4] 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.

[5] C. E. Heinbaugh, E. Luz-Ricca, and H. Shao, "Data-free one-shot federated learning under very high statistical heterogeneity," in *The Eleventh International Conference on Learning Representations*, 2023.

[6] C. Thapa, M. A. P. Chamikara, and S. A. Camtepe, *Advancements of Federated Learning Towards Privacy Preservation: From Federated Learning to Split Learning*. Cham: Springer International Publishing, 2021, pp. 79–109.

[7] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtep, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for internet of things," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*, 2020, pp. 91–100.

[8] Z. Lin, G. Qu, X. Chen, and K. Huang, "Split learning in 6g edge networks," *IEEE Wireless Communications*, pp. 1–7, 2024.

[9] J. Jeon and J. Kim, "Privacy-sensitive parallel split learning," in *2020 International Conference on Information Networking (ICOIN)*. IEEE, 2020, pp. 7–9.

[10] R. Taylor, "Inside the AI care home: the smart tech making old people safer," *The Sunday Times*, Nov. 2024.

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

[12] Z. Li, W. Wu, S. Wu, and W. Wang, "Adaptive split learning over energy-constrained wireless edge networks," 2024.

[13] M. Zhang, J. Cao, Y. Sahni, X. Chen, and S. Jiang, "Resource-efficient parallel split learning in heterogeneous edge computing," *arXiv preprint arXiv:2403.15815*, 2024.

[14] E. Erdoğan, A. Küpçü, and A. E. Çiçek, "Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning," in *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, 2022, pp. 115–124.

[15] Sinha *et al.*, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2018.

[16] W. Yao, C. Yu, S. Zeng, and J. Zhang, "Constrained bi-level optimization: Proximal lagrangian value function approach and hessian-free algorithm," *ArXiv*, vol. abs/2401.16164, 2024.

[17] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment," *IEEE Transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, 2011.

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

[19] D. Pasquini, G. Ateniese, and M. Bernaschi, "Unleashing the tiger: Inference attacks on split learning," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2113–2129.

[20] S. A. Khowaja, I. H. Lee, K. Dev, M. A. Jarwar, and N. M. F. Qureshi, "Get your foes fooled: Proximal gradient split learning for defense against model inversion attacks on iomt data," *IEEE Transactions on Network Science and Engineering*, 2022.

[21] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith, and C. Wang, "Label leakage and protection in two-party split learning," in *International Conference on Learning Representations*, 2022.

[22] T. Titcombe, A. J. Hall, P. Papadopoulos, and D. Romanini, "Practical defences against model inversion attacks for split neural networks," *ICLR Workshop on Distributed and Private Machine Learning (DPML)*, 2021.

[23] N. Haim, G. Vardi, G. Yehudai, O. Shamir, and M. Irani, "Reconstructing training data from trained neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 911–22 924, 2022.

[24] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9*, 2015.

[26] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[27] N. Corporation, "Jetson nano - powerful ai at your edge." [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/

[28] E.-G. Talbi, *A Taxonomy of Metaheuristics for Bi-level Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–39.

[29] I. Kasa Companies, "python-kasa: Python api for tp-link smarthome products."

[30] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *ArXiv*, vol. abs/1708.07747, 2017.

[31] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[32] M. U. Rehman, I. Uddin, M. Adnan, A. Tariq, and S. Malik, "Vta-smac: Variable traffic-adaptive duty cycled sensor mac protocol to enhance overall qos of s-mac protocol," *IEEE Access*, vol. 9, pp. 33 030–33 040, 2021.

[33] H. Yang, M. Ge, D. Xue, K. Xiang, H. Li, and R. Lu, "Gradient leakage attacks in federated learning: Research frontiers, taxonomy, and future directions," *IEEE Network*, vol. 38, no. 2, pp. 247–254, 2024.

[34] X. Liu, S. Cai, Q. Zhou, S. Guo, R. Li, and K. Lin, "Gradient diffusion: A perturbation-resilient gradient leakage attack," *CoRR*, vol. abs/2407.05285, 2024.

[35] Z. Zhao, D. Liu, Y. Cao, T. Chen, S. Zhang, and H. Tang, "U-shaped split federated learning: An efficient cross-device learning framework with enhanced privacy-preserving," in *2023 9th International Conference on Computer and Communications (ICCC)*, 2023, pp. 2182–2186.