ONE-SHOT SPLIT FEDERATED LEARNING WITH PROTOTYPE-GUIDED SYNTHETIC SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Split Federated Learning (SFL) enables collaborative training across resource-constrained clients by partitioning models between clients and the server. However, under strict communication constraints, reducing client-server interaction to a single round severely limits available supervision, making effective training challenging under heterogeneous and imbalanced data distributions. In this work, we investigate one-shot SFL and identify the lack of reliable global supervision from limited intermediate representations as the key challenge. Existing one-shot federated learning methods typically assume access to complete local models or full model aggregation, making them difficult to apply to split federated learning where clients only optimize partial models. To address this issue, we propose Heterogeneous One-shot Split Federated Learning (HOSFL), a framework that constructs class-aware supervision from one-shot uploaded representations. HOSFL aggregates cross-client class prototypes and employs prototype-guided feature alignment and classifier-guided class consistency constraints to generate class-balanced synthetic samples, enabling effective model optimization within a single communication round. Extensive experiments on multiple benchmarks under diverse non-IID settings demonstrate that HOSFL achieves consistent performance improvements while significantly reducing communication overhead, with an average accuracy gain of 4.72% over strong SFL baselines. The anonymous code is available at `https://anonymous.4open.science/r/HOSFL-10B3`.

1 INTRODUCTION

Federated Learning (FL) is a distributed learning paradigm that enables collaborative model training across decentralized clients without sharing raw data Kairouz et al. (2021); Wang et al. (2025), and has been widely adopted in applications such as healthcare Eden et al. (2025) and smart terminals/IoT Nguyen et al. (2021). However, in real-world deployments with resource-constrained clients, standard FL often suffers from low training efficiency, since all clients must repeatedly perform local optimization and upload full model updates, while heterogeneous compute and storage resources further exacerbate straggler effects. To mitigate these limitations, Split Federated Learning (SFL) Thapa et al. (2022) integrates the model partitioning mechanism of Split Learning (SL) Gupta & Raskar (2018) with the parallel collaboration framework of FL by splitting the model into client-side and server-side components, thereby offloading part of the computation to the server while keeping raw data local. This design reduces the client-side burden and improves scalability under resource constraints, making SFL attractive for decentralized settings such as intelligent healthcare Chen et al. (2025b), vehicular networks Dai et al. (2025), edge sensing Liao et al. (2024b), and large-scale IoT systems Liao et al. (2024a).

However, despite its advantages in resource-constrained deployments, split federated learning (SFL) faces a fundamental challenge under strict communication constraints. In particular, when client–server interaction is reduced to a single round, the amount of supervisory information that can be exchanged becomes extremely limited, making it difficult to train accurate models. This issue becomes even more severe when client data are highly heterogeneous and class-imbalanced Buda et al. (2018). As illustrated in Fig. 1, the heterogeneity considered in this work includes both missing-class scenarios, in which some clients entirely lack certain classes, and overlapping-class scenarios with highly skewed class proportions. Under such conditions, existing SFL baselines often experience

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

notable performance degradation, suggesting that one-shot communication and client-side class imbalance must be addressed jointly rather than in isolation.

A straightforward way to alleviate communication overhead is to move from conventional multi-round SFL to a one-shot paradigm. Prior studies have improved the efficiency of SFL through cluster-based parallelism Liao et al. (2024b) and selective parameter synchronization Gao et al. (2024), while class imbalance in conventional FL is commonly addressed via re-weighting Cheng et al. (2024); Wang et al. (2021) or knowledge distillation Zhang et al. (2022b); Zhu et al. (2021). However, these existing approaches do not resolve the core difficulty in one-shot SFL. Methods designed for communication-efficient SFL still rely on iterative client–server interaction, whereas imbalance-mitigation strategies in conventional FL usually assume access to complete and well-optimized local models. Under the split setting, neither assumption is readily satisfied once communication is strictly restricted to a single round.

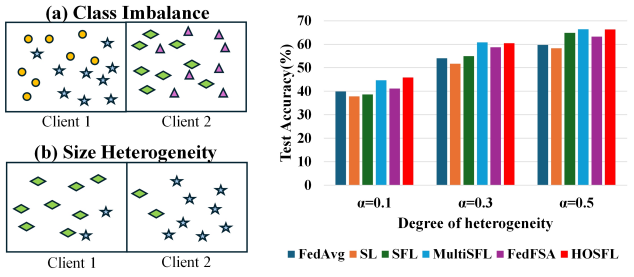

Figure 1: Illustration of heterogeneous data distributions considered in this work and accuracy comparison with multi-round baselines on CIFAR-10.

This limitation becomes more evident in one-shot federated learning. Existing one-shot FL methods typically rely on knowledge distillation Li et al. (2020); Dai et al. (2024) or generative modeling Zhang et al. (2022a); Bai et al. (2025), assuming that clients can optimize complete local models and provide effective supervision to the server within a single round. However, this assumption does not hold in SFL, where models are partitioned between clients and the server and clients only maintain partial networks. Therefore, the key challenge of one-shot SFL lies not only in reducing communication rounds, but also in recovering useful training information from limited split-accessible representations without iterative interaction or complete local models.

To this end, we propose HOSFL, a heterogeneous one-shot split federated learning framework designed for the challenging setting where only a single client-server interaction is allowed. Unlike existing one-shot FL methods that rely on fully trained local models, HOSFL exploits the intermediate representations produced by client-side front models as the only globally available information in split learning. Based on these one-shot uploaded features, the federated server constructs a class-aware supervision mechanism by combining cross-client prototype aggregation, prototype-guided feature alignment, and classifier-guided class consistency, enabling the generation of class-balanced synthetic samples without iterative communication. Meanwhile, following SFL Thapa et al. (2022), HOSFL applies the same differential privacy noise during intermediate feature transmission. As a result, HOSFL achieves communication-efficient and privacy-preserving learning while remaining robust to severe client heterogeneity and label imbalance.

Our contributions are summarized as follows: ❶ We propose HOSFL, a one-shot split federated learning framework that completes training with a single communication round, substantially reducing communication overhead compared with conventional multi-round SFL methods; ❷ We address class imbalance in one-shot SFL by constructing split-compatible class-aware synthetic supervision from uploaded intermediate representations through prototype-guided feature alignment and classifier-guided class consistency, improving robustness under highly non-IID data distributions; ❸ Extensive experiments on multiple benchmarks and heterogeneous settings demonstrate that HOSFL consistently achieves competitive or superior accuracy compared with strong SFL baselines while maintaining significantly lower communication overhead.

2 Related Work

SFL Thapa et al. (2022) integrates the parallel training paradigm of federated learning with the model partitioning strategy of split learning, thereby reducing the computational and storage burden on clients and making it suitable for resource-constrained devices. Existing studies have extensively in-

vestigated system efficiency in SFL. Communication overhead has been reduced through compressing intermediate activations Liao et al. (2024b); Qiang et al. (2025); Singh & Adhikari (2025), decreasing feature transmission frequency Gao et al. (2024); Djuhera et al. (2025), and introducing lightweight auxiliary networks Guo et al. (2021); Lin et al. (2024; 2025). To address heterogeneity in client computation and communication conditions, prior works propose adaptive model partitioning strategies Lin et al. (2024); He et al. (2025); Fan et al. (2025), centralized server-side training schemes Mu & Shen (2025), and asynchronous buffering mechanisms Yang & Liu (2025). Despite these advances, most existing approaches still rely on multi-round communication, leaving communication overhead largely unresolved. Moreover, current studies primarily focus on system-level heterogeneity, while class imbalance in client-side data remains insufficiently addressed, particularly under one-shot communication settings.

One-shot learning typically refers to a training paradigm where only a single round of client–server interaction is performed, thereby reducing communication overhead and improving deployment efficiency. In federated learning (FL), one-shot FL has been extensively studied through knowledge distillation Kang et al. (2023); Zhang et al. (2024); Allouah et al. (2024); Li et al. (2020), generative models Zhang et al. (2022a); Yang et al. (2024); Chen et al. (2025a), data distribution fitting Zeng et al. (2024); Sang et al. (2025); Karthikeyan & Polychroniadou (2025), and global feature statistics aggregation Guan et al. (2025); Liu et al. (2024); Qi et al. (2025a). However, these approaches typically assume that clients can provide optimized complete local models or sufficient global information for supervision, which is difficult to satisfy in SFL where models are partitioned between clients and the server. Under one-shot communication constraints, clients only optimize partial models, making existing one-shot FL methods difficult to directly transfer to SFL. To the best of our knowledge, HOSFL is the first work to investigate one-shot communication in the SFL setting.

3 METHOD

3.1 TWO-STAGE REPRESENTATION LEARNING ON CLIENTS

In HOSFL, the client k only holds the client-side model (i.e., the front part of the network), denoted as $f_k(\cdot;\theta_k)$, where θ_k represents the trainable parameters. The local training dataset of client k is denoted by $D_k = \{x_i\}_{i=1}^{|D_k|}$. To make one-shot communication feasible without sacrificing representation quality under heterogeneous data, the client-side optimization is conducted via a two-stage self-supervised learning procedure: the first stage performs self-supervised representation learning solely on real data and extracts informative intermediate features for one-shot upload, while the second stage further updates the client-side model by jointly leveraging local real data and synthetic data generated by the federated server.

Self-Supervised Representation Learning and Feature Upload. To ensure that a single client upload can still provide sufficiently discriminative and transferable information for subsequent global modeling, the client first performs self-supervised training on D_k to obtain a robust representation extractor. For each sample $x \in D_k$, a multi-crop augmentation operator $A(\cdot)$ is applied to generate two global views and multiple local views: $\left\{x^{g_1}, x^{g_2}, x^{l_1}, ..., x^{l_m}\right\} = A(x)$. The client adopts the Bootstrap Your Own Latent(BYOL) framework for optimization. The online branch consists of an encoder f_k, a projector $g(\cdot)$, and a predictor $q(\cdot)$, while the target branch consists of a target encoder $\bar{f}_k$ and a target projector $\bar{g}(\cdot)$. The target branch parameters are updated via an exponential moving average (EMA) of the online branch parameters and are not updated through backpropagation. Given an input view x, the representations produced by the online and target branches are defined as in Eq. (1),

$$z = g\left(GAP\left(f_k\left(x\right)\right)\right), p = q\left(norm\left(z\right)\right), \bar{z} = \bar{g}\left(GAP\left(\bar{f}_k\left(x\right)\right)\right) \tag{1}$$

where $GAP(\cdot)$ denotes global average pooling and $norm(\cdot)$ indicates feature normalization. Based on two global views obtained from data augmentation, the client optimizes the BYOL alignment objective given in Eq. (2),

$$\mathcal{L}_{byol} = \left(2 - 2\cos\left(p_{g_1}, sg\left(\bar{z}_{g_2}\right)\right)\right) + \left(2 - 2\cos\left(p_{g_2}, sg\left(\bar{z}_{g_1}\right)\right)\right) \tag{2}$$

where $sg(\cdot)$ denotes the stop-gradient operator applied to the target branch. To further enforce local consistency, an additional alignment loss between local views and the global target representation is

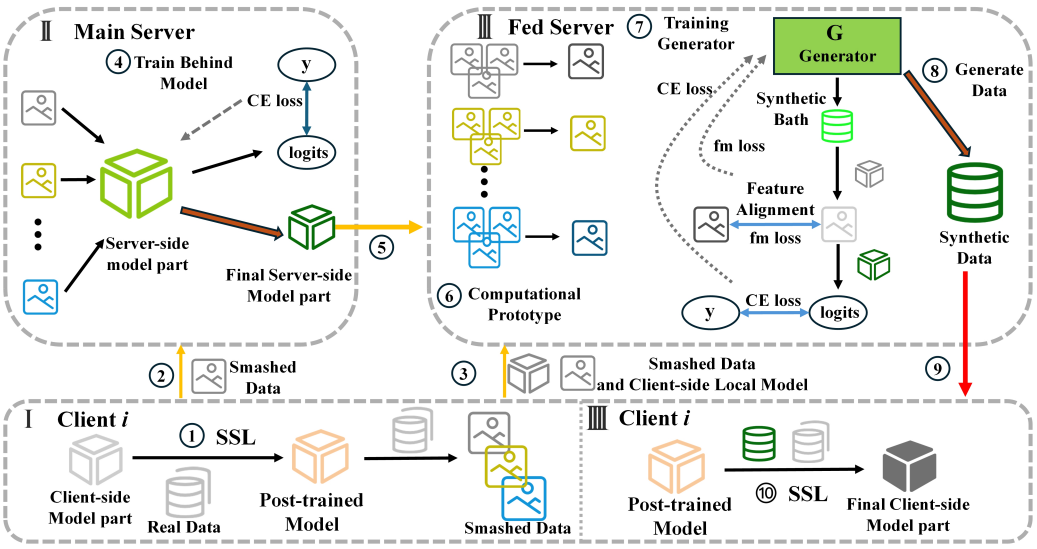

Figure 2: Illustration of the **HOSFL** framework. The pipeline consists of four stages: Stage I, clients perform self-supervised learning to obtain updated client-side front models and upload differentially private smashed data; Stage II, the main server trains a unified server-side tail model using the uploaded smashed data; Stage III, the federated server integrates uploaded smashed data, client-side model information, and the trained server-side model to capture global class structure and guide class-balanced synthetic data generation; Stage IV, the synthetic data are sent back to clients to refine the front models, alleviating missing or under-represented classes without iterative communication. The detailed procedure is summarized in Algorithm 1.

introduced, as formulated in Eq. (3).

$$\mathcal{L}_{local} = \frac{1}{m}\sum_{j=1}^{m}\left(2 - 2\left(\cos p_{l_j}, sg\left(\bar{z}_g\right)\right)\right) \tag{3}$$

To prevent representation collapse, we further impose a VICReg-style covariance regularization on the projected features: $\mathcal{L}_{\text{cov}} = \frac{1}{d(d-1)}\sum_{i\neq j}\left(\frac{1}{B-1}\sum_{b=1}^{B}(z_{b,i}-\bar{z}_i)(z_{b,j}-\bar{z}_j)\right)^2$. Combining all the above objectives, the overall loss for the first-stage self-supervised training is given in Eq. (4).

$$\mathcal{L}_{ssl1} = \mathcal{L}_{byol} + \mathcal{L}_{local} + \mathcal{L}_{cov} \tag{4}$$

The target branch parameters are updated as an EMA of the online branch parameters, $\bar{\theta} \leftarrow m\bar{\theta} + (1-m)\theta$, where m is the momentum coefficient and can be scheduled over training iterations. After the first-stage training, each client uses the updated encoder to extract intermediate feature representations from its local dataset, forming a feature set $F_k = \{(f_k(x), i_x) \mid x \in D_k\}$, where i_x denotes the sample identifier of x. The feature set F_k is uploaded to the main server for server-side optimization, while F_k and the client-side model parameters θ_k are simultaneously sent to the federated server for prototype construction and generative modeling. The uploaded sample indices are further used on the server side to recover class labels for tail-model training and class-wise prototype aggregation; detailed implementation is provided in Appendix A.

Self-Supervised Update with Synthetic Data. To compensate for missing or under-represented classes under the one-shot communication constraint, after the federated server completes generator training, each client receives the generated synthetic samples and combines them with local real data for the second-stage training process. To stabilize model updates, the client introduces a frozen encoder f_k^{old} obtained after the first-stage training as a representation anchor. A feature alignment regularization term is imposed on real samples to constrain representation drift, as formulated in Eq. (5).

$$\mathcal{L}_{feat} = \mathbb{E}_{x\sim D_k}\left\|f_k\left(x\right) - f_k^{old}\left(x\right)\right\|_2^2 \tag{5}$$

Meanwhile, the generated synthetic samples provide additional class-aware guidance through a classification objective. Specifically, a standard cross-entropy loss is applied to synthetic samples to encourage the updated model to preserve the class information contained in the generated data, as defined in Eq. (6). In practice, this objective is implemented using a lightweight local classification head attached to the client-side representation, as detailed in Appendix A.

$$\mathcal{L}_{ce} = \mathbb{E}_{(x^{fake},y)} \left[-\log p \left(y | x^{fake} \right) \right] \tag{6}$$

By jointly considering real-data-driven self-supervised learning and synthetic-data-based auxiliary constraints, the client-side model is further updated in the second training stage. Specifically, the overall optimization objective is formulated as:

$$\mathcal{L}_{ssl2} = \mathcal{L}_{ssl1}\left(real\right) + \lambda_{feat}\mathcal{L}_{feat} + \lambda_{ce}\mathcal{L}_{ce}\left(fake\right) \tag{7}$$

where $\mathcal{L}_{ssl1}(\text{real})$ denotes that the BYOL-based self-supervised objective is evaluated only on real samples, while synthetic samples are introduced through auxiliary classification and feature regularization terms. After completing the two-stage self-supervised learning procedure, each client obtains an updated client-side model for subsequent split inference and system evaluation.

3.2 Unified Tail Optimization from Smashed Features

In HOSFL, the main server maintains a single shared server-side model, denoted as $s(\cdot;\phi)$, to capture global class-discriminative information from one-shot uploaded smashed features under heterogeneous client distributions. After the first stage of self-supervised training, each client uploads its intermediate features to the main server for supervised optimization of the server-side model.

Specifically, the main server performs classification optimization on the uploaded features and their corresponding labels. To improve robustness under Non-IID data, the server-side model is optimized with a temperature-scaled cross-entropy loss, as defined in Eq. (8). Maintaining a single shared server-side model enables consistent classification learning over aggregated features from heterogeneous clients.

$$\mathcal{L}_{ce} = \frac{1}{|F|} \sum_{i} CE\left(o_T^{(i)}, y_i \right) \tag{8}$$

Here, $o_T^{(i)}$ denotes the temperature-scaled prediction logits. Temperature scaling prevents overly confident predictions when optimizing on heterogeneous client features, thereby improving training stability under distribution shifts.

In addition, to improve numerical stability and suppress excessively large prediction magnitudes, an L_2 regularization term is imposed on the temperature-scaled logits: $\mathcal{L}_{\text{logit}} = \frac{\lambda_{\text{logit}}}{2} \cdot \frac{1}{B} \sum_{i=1}^{B} \| o_T^{(i)} \|_2^2$. The overall optimization objective of the main server is formulated as:

$$\min_{\phi} \mathcal{L}_{\text{tail}} = \mathcal{L}_{ce} + \mathcal{L}_{logit} \tag{9}$$

Through this classification optimization process, the main server obtains an updated server-side model that captures global class-related feature information from heterogeneous clients. The learned model is subsequently uploaded to the federated server and used to provide classifier-guided constraints during the following generative modeling stage.

3.3 Prototype-Guided Class-Aware Global Generation

In HOSFL, the federated server constructs a generative model to alleviate client-side class imbalance without accessing raw client data. Under the one-shot communication constraint, the federated server utilizes the uploaded intermediate representations, client-side front models, and server-side tail model to construct class-aware guidance for generating balanced synthetic samples. By exploiting cross-client feature distributions and the global classification capability learned by the server-side model, HOSFL generates auxiliary samples to compensate for missing or under-represented classes.

Class Prototype Construction. To extract class-level feature information from fragmented one-shot representations, the federated server first constructs class prototypes based on the intermediate

features uploaded by clients. After the first-stage self-supervised training, clients upload extracted features together with global sample indices. Since the corresponding labels are maintained by the federated server, the server can associate uploaded features with their class labels and perform class-wise aggregation. For each class c, the corresponding prototype is computed as the mean of all associated intermediate features, as defined in Eq. (10),

$$P_c = \frac{1}{|S_c|} \sum_{f(x) \in S_c} f(x) \tag{10}$$

where S_c denotes the set of intermediate features belonging to class c. The resulting prototype characterizes the average feature distribution of class c in the shared representation space and provides a class-level reference for subsequent generator optimization.

Conditional Generator and Feature Alignment Constraint. To enable class-balanced generation under heterogeneous client distributions, the federated server first constructs a unified reference encoder by aggregating client-side front model parameters. The resulting encoder $E_{ref}(\cdot)$ is used only for feature extraction and remains frozen during generator training, providing a stable representation space for synthetic samples. Based on $E_{ref}(\cdot)$, the federated server trains a conditional generator $G(\cdot)$, which takes random noise and class labels as inputs to generate samples of corresponding classes:

$$\hat{x}_j = G(z_j, y_j).$$

The generated samples are processed using the same preprocessing operator $Prep(\cdot)$ as the client-side self-supervised training and then mapped into the representation space:

$$\hat{h}_j = E_{ref}(Prep(\hat{x}_j)).$$

To align generated features with the corresponding real feature distribution, a feature alignment constraint is introduced to pull generated representations toward their class prototypes. To preserve intra-class diversity, a perturbation is further added to the prototype representation, producing the target prototype:

$$\tilde{P}_{y_j} = P_{y_j} + \alpha \cdot \sigma(P_{y_j}) \odot \xi_j, \quad \xi \sim \mathcal{N}(0, I) \tag{11}$$

where α controls the perturbation magnitude and $\sigma(\cdot)$ represents the adaptive scaling factor. Based on the perturbed prototype, the feature alignment loss is defined as:

$$\mathcal{L}_{fm} = \frac{1}{B} \sum_{j=1}^{B} \left\| \hat{h}_j - \tilde{P}_{y_j} \right\|_2^2 \tag{12}$$

which encourages generated samples to follow the class-specific feature distributions observed from client data.

Classifier-Guided Class Consistency. Besides prototype-based feature alignment, the federated server further introduces a classifier-guided constraint to ensure that generated samples maintain their target class identities. Specifically, the server-side model parameters uploaded from the main server are combined with the reference client-side model to provide classification capability for generated samples. The corresponding classifier, denoted as $H_{global}(\cdot)$, remains frozen during generator training. The generated features $\hat{h}_j$ are fed into this classifier, and a cross-entropy loss is applied between the predicted class and the target label:

$$\mathcal{L}_{cls} = \frac{1}{B} \sum_{j=1}^{B} CE(H_{global}(\hat{h}_j), y_j) \tag{13}$$

By jointly optimizing prototype-based feature alignment and classifier-guided class consistency, the generator objective on the federated server is formulated as:

$$\min_G \mathcal{L}_G = \mathcal{L}_{fm} + \mathcal{L}_{cls} \tag{14}$$

After training, the generator produces class-balanced synthetic samples that are distributed to clients and combined with local real data for second-stage self-supervised refinement. Through this process, the feature representations of minority classes are enriched, alleviating client-side class imbalance without introducing additional communication rounds.

4 EXPERIMENTAL RESULTS

4.1 IMPLEMENTATION DETAILS

Dataset. Our evaluation experiments utilize three widely used datasets: CIFAR-10, CIFAR-100Krizhevsky et al. (2009) and SVHNNetzer et al. (2011). These datasets provide diverse benchmarks for evaluating FL performance.

Client Data Distribution. To model client-side data heterogeneity and class imbalance, we partition the training data among clients using a Dirichlet distribution with concentration parameter α, where smaller values correspond to more skewed class distributions. As shown in Fig. 3, experiments are conducted under three heterogeneity levels, $\alpha \in \{0.3, 0.1\}$, covering moderate to highly heterogeneous settings. In the visualization, each circle's size corresponds to the number of samples associated with a specific class, and circles with the same color denote the same class label. In addition, results under an extreme heterogeneity setting are reported in Appendix C.2, where our method consistently outperforms all baselines.

Baseline. To evaluate the effectiveness of HOSFL, we compare it with FedAvg McMahan et al. (2017), SL Gupta & Raskar (2018), SFL Thapa et al. (2022), FedFSA Qi et al. (2025b), and MultiSFL Xia et al. (2025). FedAvg, SL, and SFL represent standard federated, split, and split federated learning paradigms, respectively. FedFSA is adapted to the split federated learning setting with the same model architecture for fair comparison, as it originally assumes complete local models. MultiSFL is a representative SFL method targeting data heterogeneity but does not explicitly address class imbalance. Besides their original multi-round settings, we further evaluate all methods under a single communication round to assess their applicability in the challenging one-shot federated learning scenario.

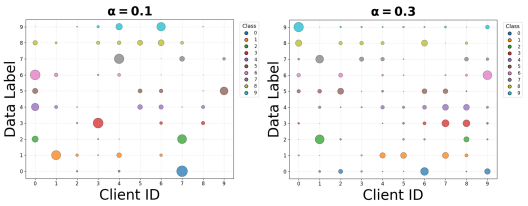

Figure 3: Visualization of heterogeneous client data distributions characterized by three diverse α values, with CIFAR-10 as illustration.

Training setting. Following conventional SFL settings, we conduct experiments with ten clients under the SplitFed framework. ResNet-18 is adopted as the unified backbone and split at the same position for all split-based methods, where the client-side front model extracts intermediate features and the server-side tail model completes the remaining computation. For fair comparison, all multi-round baselines are trained for $E = 150$ communication rounds unless otherwise specified, whereas HOSFL follows the one-shot protocol with only a single round of feature upload and synthetic-supervision feedback. FedAvg is implemented with a complete, unsplit ResNet-18 model. In the second stage of HOSFL, pseudo data are mixed with real data at a ratio of $ratio_{fake} = 0.5$, while feature alignment and pseudo supervision are introduced as regularization terms with weights $\lambda_{feat} = 0.1$ and $\lambda_{ce} = 0.3$, respectively. Additional implementation details, including label recovery, the local head for synthetic supervision, and the exact stage-wise training procedure, are provided in Appendix A. Furthermore, Appendix C.3 presents additional experiments investigating the impact of different numbers of clients on the performance of HOSFL, while Appendix C.4 provides a detailed analysis of the resource consumption required by clients.

4.2 EVALUATION

Performance Comparison. Table 1 reports the average test accuracy over multiple runs on CIFAR-10, CIFAR-100, and SVHN under different heterogeneity levels controlled by the Dirichlet parameter α. As α decreases, all methods suffer performance degradation due to increasingly heterogeneous and imbalanced client data distributions. Nevertheless, HOSFL maintains stable performance with only a single communication round, demonstrating its effectiveness in addressing severe class imbalance under communication constraints.

When conventional multi-round approaches are restricted to the one-shot setting, their performance drops significantly across different datasets and heterogeneity levels, indicating that these methods heavily rely on iterative client–server interactions to gradually obtain effective optimization signals.

Table 1: Test accuracy of different methods under multi-round and one-shot communication settings on various datasets with different heterogeneity levels α.

Dataset	Hetero. Settings	Multi-round Acc (%)					One-shot Acc (%)			
		FedAvg	SL	SFL	MultiSFL	FedFSA	SFL	MultiSFL	FedFSA	HOSFL (Ours)
SVHN	$\alpha = 0.1$	76.37	73.43	61.33	84.72	78.73	12.41	12.43	10.58	**86.76**
	$\alpha = 0.3$	80.23	76.85	73.74	88.11	84.45	17.77	19.58	17.21	**90.70**
	$\alpha = 0.5$	83.74	77.93	84.76	88.77	85.41	18.06	20.31	19.34	**91.62**
CIFAR-10	$\alpha = 0.1$	40.07	39.87	40.71	44.71	42.80	10.03	9.61	10.00	**46.87**
	$\alpha = 0.3$	52.76	51.68	55.54	60.82	58.74	12.04	13.68	12.29	**61.90**
	$\alpha = 0.5$	55.61	53.31	64.53	66.39	62.70	13.10	16.31	14.41	**67.67**
CIFAR-100	$\alpha = 0.1$	26.74	28.63	25.60	29.11	28.63	1.59	1.07	1.27	**33.64**
	$\alpha = 0.3$	28.10	31.94	28.72	32.68	31.94	1.42	1.16	1.21	**35.57**
	$\alpha = 0.5$	29.73	33.73	32.48	34.06	33.73	1.37	2.27	2.13	**37.89**

In contrast, HOSFL is specifically designed for one-shot split federated learning and constructs class-aware supervision from one-shot uploaded intermediate representations, enabling robust training without additional communication rounds. The detailed performance analysis under different client scales and heterogeneity settings is provided in Appendix C.3.

Time complexity. Fig. 4 compares the computational time cost of SFL, MultiSFL, FedFSA, and HOSFL across multiple independent runs. SFL requires repeated client–server forward and backward interactions during training, resulting in moderate runtime, while MultiSFL introduces additional multi-model maintenance and knowledge replay, leading to higher computational overhead. FedFSA further incurs extra cost from feature-space alignment and calibration operations under the split architecture. In contrast, HOSFL completes training and synthetic supervision construction within a single communication round, avoiding repeated feature exchanges and split interactions. These results show that HOSFL achieves lower computational cost while maintaining effective performance under heterogeneous data distributions.

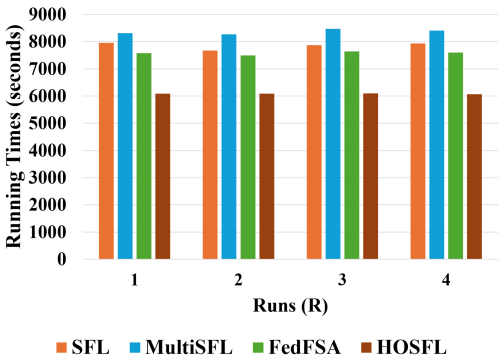

Figure 4: Time complexity comparison between SFL, MultiSFL, FedFSA and HOSFL based on four repeated runs (seconds) on CIFAR10.

Sensitivity analysis of key second-stage hyperparameters. We investigate the sensitivity of three hyperparameters in the second-stage training: λ_{feat}, λ_{ce}, and $ratio_{fake}$. As shown in Fig. 5, each experiment varies one parameter while fixing the others, and all parameters exhibit unimodal trends. Specifically, $\lambda_{\text{feat}} \approx 0.1$ achieves the best trade-off by constraining representation drift without excessive regularization. Similarly, $\lambda_{\text{ce}} \approx 0.3$ provides effective class-aware guidance from synthetic samples, while overly weak or strong constraints reduce its benefit. For $ratio_{fake}$, incorporating synthetic data consistently improves performance, with the best result achieved around $ratio_{fake} = 0.5$. These results confirm that synthetic data provide complementary class-aware guidance under the one-shot constraint rather than replacing real-data-driven self-supervised learning.

Ablation Study. Table 2 presents the ablation results across different datasets and heterogeneity levels. We evaluate two variants: HOSFL w/o SSL1, which removes the first-stage client-side self-supervised learning, and HOSFL w/o FS, which removes server-side generative modeling and the subsequent client refinement. Removing SSL1 consistently decreases performance, demonstrating the importance of learning robust client representations before one-shot aggregation. A more significant degradation is observed when server-side generation and refinement are removed, particularly under highly heterogeneous settings, indicating that synthetic supervision is essential for alleviating class imbalance and missing-class effects. Overall, the complete HOSFL benefits from the complementary roles of self-supervised representation learning and synthetic-data-assisted refinement.

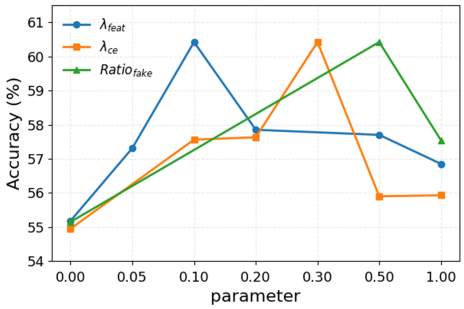

Figure 5: Sensitivity analysis of second-stage hyperparameters λ_{feat}, λ_{ce}, and $ratio_{fake}$ on CIFAR-10 under label heterogeneity ($\alpha = 0.3$).

Table 2: Ablation study under different datasets and heterogeneity settings α.

Dataset	α	HOSFL w/o SSL1	HOSFL w/o FS	HOSFL Full Model
SVHN	α=0.1	38.34	60.72	**86.76**
	α=0.3	59.12	81.66	**90.70**
	α=0.5	65.03	87.48	**91.62**
CIFAR-10	α=0.1	13.12	35.28	**46.87**
	α=0.3	16.22	51.96	**61.90**
	α=0.5	20.21	61.94	**67.67**
CIFAR-100	α=0.1	4.47	24.49	**33.64**
	α=0.3	10.11	31.54	**37.57**
	α=0.5	14.04	33.23	**37.89**

Privacy Evaluation. We evaluate the privacy risks of HOSFL from both intermediate feature transmission and synthetic data generation perspectives. For a fair comparison, HOSFL and conventional SFL adopt the same differential privacy mechanism during intermediate feature transmission, allowing the evaluation to focus on the impact of the communication paradigm. As shown in Fig. 6, HOSFL achieves lower PSNR and SSIM values than conventional SFL, indicating weaker feature reconstructability and reduced leakage risk under the same privacy protection setting. Furthermore, Fig. 7 shows that the generated synthetic samples are visually distinct from the original training images, suggesting that HOSFL does not memorize client data during generation. The detailed analysis of privacy leakage evaluation and synthetic data visualization is provided in Appendix C.5. Overall, HOSFL reduces potential privacy risks caused by repeated feature exchanges while maintaining effective knowledge transfer under the one-shot communication constraint.

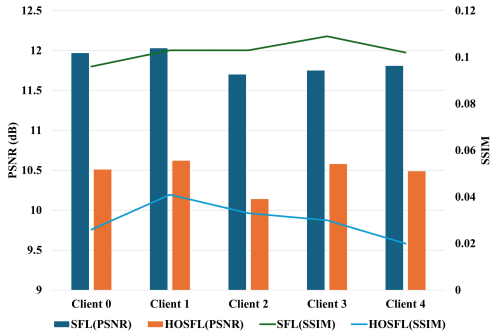

Figure 6: Privacy comparison between HOSFL and SFL under heterogeneous CIFAR-10 data ($\alpha = 0.3$).



Figure 7: Visualization of synthetic data generated by the generator on CIFAR-10.

5 Conclusion

We studied split federated learning under a one-shot communication constraint and proposed **HOSFL**, a framework that enables effective cross-client learning without iterative feature exchanges. By aggregating intermediate representations to construct class prototypes and training a conditional generator with prototype-guided feature alignment and classifier-guided class consistency, HOSFL generates class-balanced synthetic samples to alleviate class imbalance and missing-class issues. A two-stage client-side self-supervised strategy further stabilizes representation refinement while preserving the one-shot constraint. Extensive experiments across multiple datasets and heterogeneous settings demonstrate that HOSFL achieves strong performance with reduced communication and runtime costs, particularly under severe non-IID conditions, highlighting its effectiveness for communication-constrained split federated learning.

AI USE STATEMENT

Generative AI tools were used in this work for language polishing, grammar checking, and improving the readability of the manuscript. We did not use generative AI tools to generate research ideas, design the proposed method, conduct experiments, analyze experimental results, or produce scientific claims. All AI-assisted modifications were manually reviewed and verified by the authors. The authors take full responsibility for the final content of this work, including all text, experimental results, and technical contributions.

ETHICS STATEMENT

This work studies communication-efficient federated learning using publicly available benchmark datasets. No private user data or personally identifiable information is collected or processed in our experiments. Although we evaluate potential privacy leakage risks from intermediate representations and synthetic data generation, the evaluation is conducted only on public datasets and does not involve real-world user information. We follow responsible practices in designing and evaluating federated learning systems.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architecture, hyperparameter configurations, data partition strategies, and experimental protocols in the paper and appendix. All experiments are conducted on publicly available benchmark datasets, including CIFAR-10, CIFAR-100, and SVHN. The released implementation contains the training scripts and evaluation procedures required to reproduce the reported results.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Youssef Allouah, Akash Dhasade, Rachid Guerraoui, Nirupam Gupta, Anne-Marie Kermarrec, Rafael Pinot, Rafael Pires, and Rishi Sharma. Revisiting ensembling in one-shot federated learning. *Advances in Neural Information Processing Systems*, 37:68500–68527, 2024.

Jun Bai, Yiliao Song, Di Wu, Atul Sajjanhar, Yong Xiang, Wei Zhou, Xiaohui Tao, Yan Li, and Yue Li. A unified solution to diverse heterogeneities in one-shot federated learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 71–82, 2025.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks*, 106:249–259, 2018.

Haokun Chen, Hang Li, Yao Zhang, Jinhe Bi, Gengyuan Zhang, Yueqi Zhang, Philip Torr, Jindong Gu, Denis Krompass, and Volker Tresp. Fedbip: Heterogeneous one-shot federated learning with personalized latent diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 30440–30450, 2025a.

Yishan Chen, Xiangwei Zeng, Huashuai Cai, Qing Xu, and Zhiquan Liu. Decentralized qos-aware model inference using federated split learning for cloud-edge medical detection. *IEEE Transactions on Parallel and Distributed Systems*, 2025b.

Xu Cheng, Fan Shi, Yao Zhang, Huan Li, Xiufeng Liu, and Shengyong Chen. Frame: Feature rectification for class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Cheng Dai, Tianli Zhu, Sha Xiang, Lipeng Xie, Sahil Garg, and M Shamim Hossain. Psfl: Personalized split federated learning framework for distributed model training in intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

Rong Dai, Yonggang Zhang, Ang Li, Tongliang Liu, Xun Yang, and Bo Han. Enhancing one-shot federated learning through data and ensemble co-boosting. *arXiv preprint arXiv:2402.15070*, 2024.

Aladin Djuhera, Vlad C. Andrei, Xinyang Li, Ullrich J. Mönich, Holger Boche, and Walid Saad. R-sfllm: Jamming resilient framework for split federated learning with large language models. *IEEE Transactions on Information Forensics and Security*, 20:8296–8311, 2025. doi: 10.1109/TIFS.2025.3594107.

Rebekah Eden, Ignatius Chukwudi, Chris Bain, Sebastiano Barbieri, Leonie Callaway, Susan de Jersey, Yasmeen George, Alain-Dominique Gorse, Michael Lawley, Peter Marendy, et al. A scoping review of the governance of federated learning in healthcare. *npj Digital Medicine*, 8(1):427, 2025.

Wenhao Fan, Penghui Chen, Xiongfei Chun, and Yuan'an Liu. Madrl-based model partitioning, aggregation control, and resource allocation for cloud-edge-device collaborative split federated learning. *IEEE Transactions on Mobile Computing*, 2025.

Yunqi Gao, Bing Hu, Mahdi Boloursaz Mashhadi, Wei Wang, and Mehdi Bennis. Pipesfl: A fine-grained parallelization framework for split federated learning on heterogeneous clients. *IEEE transactions on mobile computing*, 2024.

Zenghao Guan, Yucan Zhou, and Xiaoyan Gu. Capture global feature statistics for one-shot federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 16942–16950, 2025.

Jialin Guo, Jie Wu, Anfeng Liu, and Neal N Xiong. Lightfed: An efficient and secure federated edge learning system on model splitting. *IEEE Transactions on Parallel and Distributed Systems*, 33 (11):2701–2713, 2021.

Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

Qiang He, Kaibin Wang, Zeqian Dong, Liang Yuan, Feifei Chen, Hai Jin, and Yun Yang. Hourglass: Enabling efficient split federated learning with data parallelism. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1317–1333, 2025.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

Myeongkyun Kang, Philip Chikontwe, Soopil Kim, Kyong Hwan Jin, Ehsan Adeli, Kilian M Pohl, and Sang Hyun Park. One-shot federated learning on medical data using knowledge distillation with image synthesis and client model adaptation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 521–531. Springer, 2023.

Harish Karthikeyan and Antigoni Polychroniadou. Opa: one-shot private aggregation with single client interaction and its applications to federated learning. In *Annual International Cryptology Conference*, pp. 319–353. Springer, 2025.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. *arXiv preprint arXiv:2010.01017*, 2020.

Yunming Liao, Yang Xu, Hongli Xu, Lun Wang, Zhiwei Yao, and Chunming Qiao. Mergesfl: Split federated learning with feature merging and batch size regulation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 2054–2067. IEEE, 2024a.

Yunming Liao, Yang Xu, Hongli Xu, Zhiwei Yao, Liusheng Huang, and Chunming Qiao. Parallelsfl: A novel split federated learning framework tackling heterogeneity issues. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 845–860, 2024b.

Zheng Lin, Guanqiao Qu, Wei Wei, Xianhao Chen, and Kin K Leung. Adaptsfl: Adaptive split federated learning in resource-constrained edge networks. *arXiv preprint arXiv:2403.13101*, 2024.

Zheng Lin, Wei Wei, Zhe Chen, Chan-Tong Lam, Xianhao Chen, Yue Gao, and Jun Luo. Hierarchical split federated learning: Convergence analysis and system optimization. *IEEE Transactions on Mobile Computing*, 24(10):9352–9367, 2025. doi: 10.1109/TMC.2025.3565509.

Xiang Liu, Liangxi Liu, Feiyang Ye, Yunheng Shen, Xia Li, Linshan Jiang, and Jialin Li. Fedlpa: One-shot federated learning with layer-wise posterior aggregation. *Advances in Neural Information Processing Systems*, 37:81510–81548, 2024.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Yujia Mu and Cong Shen. Federated split learning with improved communication and storage efficiency. *IEEE Transactions on Mobile Computing*, 2025.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 7. Granada, 2011.

Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE communications surveys & tutorials*, 23(3):1622–1658, 2021.

Fan Qi, Ziqi Zhao, Zixin Zhang, and Huaiwen Zhang. One-shot multimodal federated learning via diverse synthetic feature optimization. In *Proceedings of the 7th ACM International Conference on Multimedia in Asia*, pp. 1–11, 2025a.

Zhuang Qi, Lei Meng, Zhaochuan Li, Han Hu, and Xiangxu Meng. Cross-silo feature space alignment for federated learning on clients with imbalanced data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19986–19994, 2025b.

Xianke Qiang, Zheng Chang, Chaoxiong Ye, Timo Hämäläinen, and Geyong Min. Split federated learning empowered vehicular edge intelligence: Concept, adaptive design, and future directions. *IEEE Wireless Communications*, 32(4):90–97, 2025. doi: 10.1109/MWC.009.2400219.

Tian Sang, Zhiguang Chu, Jiang Xuan, Xing Zhang, and Xiang Li. Personalized federated learning in one-shot: A method for heterogeneous data scenarios. *IEEE Internet of Things Journal*, 2025.

Neha Singh and Mainak Adhikari. A hybrid semi-asynchronous federated learning and split learning strategy in edge networks. *IEEE Transactions on Network Science and Engineering*, 12(2): 1429–1439, 2025. doi: 10.1109/TNSE.2025.3530999.

Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 8485–8493, 2022.

Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Addressing class imbalance in federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10165–10173, 2021.

Zheng Wang, Zihui Wang, Xiaoliang Fan, and Cheng Wang. Federated learning with domain shift eraser. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 4978–4987, 2025.

Zeke Xia, Ming Hu, Dengke Yan, Ruixuan Liu, Anran Li, Xiaofei Xie, and Mingsong Chen. Multisfl: Towards accurate split federated learning via multi-model aggregation and knowledge replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 914–922, 2025.

Jiarong Yang and Yuan Liu. Gas: Generative activation-aided asynchronous split federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 21956–21964, 2025.

Mingzhao Yang, Shangchao Su, Bin Li, and Xiangyang Xue. Feddeo: Description-enhanced one-shot federated learning with diffusion models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 6666–6675, 2024.

Hui Zeng, Minrui Xu, Tongqing Zhou, Xinyi Wu, Jiawen Kang, Zhiping Cai, and Dusit Niyato. One-shot-but-not-degraded federated learning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 11070–11079, 2024.

Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. Dense: Data-free one-shot federated learning. *Advances in Neural Information Processing Systems*, 35:21414–21428, 2022a.

Junyuan Zhang, Songhua Liu, and Xinchao Wang. One-shot federated learning via synthetic distiller-distillate communication. *Advances in Neural Information Processing Systems*, 37:102611–102633, 2024.

Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10174–10183, 2022b.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pp. 12878–12889. PMLR, 2021.

A Expanded Algorithm and Implementation Clarifications

This appendix provides an expanded (reproducible) description of HOSFL under the one-shot split federated learning (SFL) constraint. We strictly follow the notations and definitions in the main paper, including $f_k(\cdot;\theta_k)$, $s(\cdot;\phi)$, P_c and $\tilde{P}_{y_j}$, E_{ref}, H_{global}, and Eq.(4)–Eq.(14).

A.1 Expanded One-shot HOSFL Procedure

One-Shot Training Feasibility. After the one-shot uploads of F_k and θ_k from the clients, together with ϕ from the main server, the federated server is able to train G without requiring any further client-side gradients or split interactions. This property is formalized in Proposition B.1.

A.2 Practical Clarifications Required for Reproducibility

(1) Label recovery via indices i_x. Clients upload intermediate features along with global sample indices i_x. The servers use these indices to recover the corresponding class labels, enabling: (i) supervised training of the tail model on the main server (Sec.3.2); and (ii) class-wise aggregation for prototype construction on the federated server (Sec.3.3). This mechanism avoids accessing raw data while still providing class supervision.

(2) What exactly is optimized (and what is not) during generator training. During federated-server generator training, the only trainable module is $G(\cdot)$. Both $E_{\text{ref}}(\cdot)$ (constructed by federated averaging over $\{\theta_k\}$) and $H_{\text{global}}(\cdot)$ (instantiated from the uploaded ϕ) remain frozen, exactly as described in Sec.3.3. Therefore, L_{fm} and L_{cls} are computed by forwarding generated samples through frozen modules, and no gradients flow into client-side models or the tail model during this phase.

Algorithm 1 HOSFL: Heterogeneous One-shot Split Federated Learning Framework

Require: Client datasets $\{D_k\}_{k=1}^{K}$; client-side (front) models $\{f_k(\cdot;\theta_k)\}$; main-server (tail) model $s(\cdot;\phi)$.
Ensure: Updated $\{f_k\}$ and s.

Stage I: Client-side self-supervised training and one-shot upload
for all client $k = 1, \ldots, K$ **in parallel do**
  Optimize θ_k on D_k by minimizing L_{ssl1} in Eq.4; save a frozen snapshot f_k^{old} at the end of Stage I (used in Eq.5).
  Extract intermediate features $F_k = \{(f_k(x), i_x) \mid x \in D_k\}$.
  Upload F_k to the **main server**; upload F_k and θ_k to the **federated server**.
end for
Main server: Supervised optimization of a unified tail model
Collect $\{F_k\}$ and recover labels via indices; optimize ϕ by minimizing the objective in Eq. 9, which consists of the cross-entropy loss in Eq. 8 together with the stabilization terms.
Upload the trained tail parameters ϕ to the **federated server** to instantiate H_{global}.
Federated server: Prototype construction and generator training
Construct class prototypes $\{P_c\}$ via Eq.10 from uploaded features and recovered labels.
Build the reference encoder $E_{\text{ref}}(\cdot)$ by federated averaging over $\{\theta_k\}$; keep E_{ref} **frozen** during generator training.
Instantiate $H_{\text{global}}(\cdot)$ from the uploaded ϕ; keep H_{global} **frozen** during generator training.
Train the conditional generator $G(\cdot)$ by minimizing Eq.14: $L_G = L_{fm} + L_{cls}$, where L_{fm} uses perturbed prototypes $\tilde{P}_{y_j}$ in Eq.11 and L_{cls} uses H_{global} in Eq.13.
Generate labeled synthetic samples and distribute them to each client (one-shot download).
Stage II: Client-side stable update with synthetic data
for all client $k = 1, \ldots, K$ **in parallel do**
  Update θ_k by minimizing Eq.7: $L_{ssl2} = L_{ssl1}(real) + \lambda_{feat} L_{feat} + \lambda_{ce} L_{ce}(fake)$.
  Compute L_{ssl1} and L_{feat} on real samples only (Eq. (4) and Eq. (5)), and compute L_{ce} on synthetic samples only (Eq. (6)).
end for

(3) Implementing Eq.6 on clients: a lightweight local head for labeled synthetic samples. Eq.(6) imposes a labeled constraint on synthetic samples:

$$L_{ce} = \mathbb{E}_{(x_{\text{fake}},y)}\big[-\log p(y \mid x_{\text{fake}})\big].$$

In Stage II, clients implement $p(y \mid x_{\text{fake}})$ using a lightweight local classification head $h_k(\cdot)$ (e.g., a single linear layer), applied on the online representation used by BYOL. Let z_{fake} denote the online-branch representation of a synthetic view (consistent with the representation in Eq.(1)). Then

$$p(y \mid x_{\text{fake}}) = \mathrm{Softmax}\big(h_k(z_{\text{fake}})\big), \qquad L_{ce}(fake) = \mathbb{E}_{(x_{\text{fake}},y)}\Big[\mathrm{CE}\big(h_k(z_{\text{fake}}), y\big)\Big].$$

This head is used only for training with labeled synthetic data.

(4) Real-only SSL and fake-only labeled supervision (stabilization in Stage II). To prevent low-quality synthetic samples from destabilizing self-supervised learning, Stage II uses a strict loss split:

- The self-supervised objective L_{ssl1} (Eq.(4)) is evaluated on **real samples only**.
- The anchoring term L_{feat} (Eq.(5)) is evaluated on **real samples only**.
- Synthetic samples contribute only through the labeled constraint $L_{ce}(fake)$ (Eq.(6)).

This implementation matches the design stated below Eq.7 in the main paper: synthetic data is incorporated only via auxiliary terms.

One-shot distribution of synthetic data. After training the generator on the federated server, synthetic supervision is distributed to clients in a one-shot manner. Specifically, the federated server directly generates class-balanced synthetic samples using the trained generator and prepares labeled

synthetic datasets for each client. These synthetic samples are generated entirely on the federated server and are broadcast to clients together with their corresponding labels, avoiding any additional generation or reconstruction overhead on the client side. Upon receiving the synthetic samples, each client applies the same preprocessing operator $\mathrm{Prep}(\cdot)$ as used in Stage I and incorporates them into the second-stage training. This design is fully consistent with the one-shot communication constraint and aligns with the Stage-II training objective in Eq. 7, where self-supervised learning is performed exclusively on real samples, while synthetic samples are used only through the labeled supervision term in Eq. 6.

B Analysis and Proofs

This appendix provides short, self-contained analyses and proofs that are tightly coupled to the design of HOSFL: (i) why generator training is feasible under one-shot SFL; (ii) why prototype alignment in the E_{ref} space is coherent; (iii) key properties of perturbed prototypes in Eq.11; (iv) why feature alignment and classifier-guided class consistency are complementary; (v) why the Stage-II anchor term stabilizes training.

B.1 One-Shot Dependency Sufficiency for Generator Training

Proposition B.1 (One-Shot Dependency Sufficiency). After Stage I, suppose clients perform one-shot uploads $\{F_k\}_{k=1}^K$ and $\{\theta_k\}_{k=1}^K$, and the main server uploads ϕ to the federated server. Then the generator optimization in Eq.14 can be written as

$$\min_G \; L_G\big(G; \{P_c\}, E_{\text{ref}}, H_{\text{global}}\big),$$

where $\{P_c\}$ is computed from $\{F_k\}$ by Eq.10, E_{ref} is computed by federated averaging over $\{\theta_k\}$ and is frozen during generator training, and H_{global} is instantiated from ϕ and is frozen during generator training. Hence, once these one-shot uploads are completed, optimizing G requires no additional client-server interactions.

Proof. By Eq. 10, each prototype P_c depends solely on the uploaded intermediate features and the recovered labels. By construction in Sec. 3.3, E_{ref} depends only on $\{\theta_k\}$ and is frozen during generator training. Similarly, H_{global} is instantiated from ϕ and remains frozen. Each term in Eq. 14, including L_{fm} in Eq. 12, L_{cls} in Eq. 13, is computed solely from $\{P_c\}$, E_{ref}, H_{global}, and the generator outputs. Therefore, G can be optimized entirely on the federated server without further communication rounds.

B.2 Consistency Between Prototypes and the Reference Encoder Space

In HOSFL, prototypes P_c are computed by averaging uploaded client features (Eq.10), while generated features are extracted by the frozen reference encoder E_{ref}. A natural question is whether the prototype targets and the reference-encoder features live in a compatible representation space. We provide a mild bound that explains why this is reasonable when client-side parameters do not diverge excessively.

Assumption 1 (Local sensitivity to parameters). All client-side front models share the same architecture and parameterization. There exists a bounded parameter set Θ that contains $\{\theta_k\}_{k=1}^K$ and their federated average $\bar{\theta}$. For any sample x and any two parameter vectors $\theta, \theta' \in \Theta$, there exists $L_\theta > 0$ such that

$$\|f(x;\theta) - f(x;\theta')\|_2 \le L_\theta \|\theta - \theta'\|_2. \tag{B.1}$$

Let $\bar{\theta} = \frac{1}{K}\sum_{k=1}^K \theta_k$ be the federated average, and write the reference encoder as $E_{\text{ref}}(\cdot) = f(\cdot;\bar{\theta})$ (consistent with Sec. 3.3).

Proposition 1 (Prototype consistency under federated averaging). Fix a class c and let S_c denote the set of samples whose labels equal c (as induced by indices I_k used for Eq.10). Define the "reference-space prototype"

$$\bar{P}_c = \frac{1}{|S_c|} \sum_{(k,x)\in S_c} f(x;\bar{\theta}).$$

Then

$$\|P_c - \bar{P}_c\|_2 \le \frac{L_\theta}{|S_c|} \sum_{(k,x)\in S_c} \|\theta_k - \bar{\theta}\|_2. \tag{B.2}$$

Proof. By Eq.(10),

$$P_c = \frac{1}{|S_c|} \sum_{(k,x)\in S_c} f(x;\theta_k).$$

Thus

$$P_c - \bar{P}_c = \frac{1}{|S_c|} \sum_{(k,x)\in S_c} \big(f(x;\theta_k) - f(x;\bar{\theta})\big).$$

Taking norms and applying the triangle inequality and Assumption 1 yields Eq.B.2.

Interpretation. When client-side parameters are not too dispersed (a common empirical outcome under a shared SSL objective and initialization), the prototype computed from client features is close to the corresponding prototype in the reference encoder space. This justifies aligning generated features $\hat{h} = E_{\text{ref}}(\text{Prep}(\hat{x}))$ to P_c via L_{fm}.

B.3 Feature Matching and Perturbed Prototypes

Mean–variance decomposition for feature matching (no perturbation). Fix a class c and let $x = G(z, c)$ induce a random generated feature $\hat{h} = E_{\text{ref}}(\text{Prep}(x))$. Ignoring perturbation for clarity (i.e., $\tilde{P}_c = P_c$), the expected squared distance admits the classical decomposition.

Proposition 2 (Mean–variance decomposition). If $\hat{h} \sim q_c(\hat{h})$, then

$$\mathbb{E}_{q_c}\big[\|\hat{h} - P_c\|_2^2\big] = \text{Tr}\big(\text{Cov}_{q_c}[\hat{h}]\big) + \|\mathbb{E}_{q_c}[\hat{h}] - P_c\|_2^2. \tag{B.3}$$

Proof. Let $\mu = \mathbb{E}[\hat{h}]$ and expand $\|\hat{h} - P_c\|^2 = \|\hat{h} - \mu + \mu - P_c\|^2$. Taking expectation cancels the cross term and yields Eq.B.3.

Implication. Minimizing the expected feature matching term simultaneously (i) reduces within-class dispersion in the representation space and (ii) pulls the mean generated representation toward the class prototype.

Properties of perturbed prototypes in Eq.11. Recall Eq.(11): $\tilde{P}_c = P_c + \alpha \cdot \sigma(P_c) \odot \xi$ with $\xi \sim \mathcal{N}(0, I)$.

Proposition 3 (Unbiasedness and constant shift). For any fixed vector h,

$$\mathbb{E}_\xi[\tilde{P}_c] = P_c, \tag{B.4}$$

and

$$\mathbb{E}_\xi\big[\|h - \tilde{P}_c\|_2^2\big] = \|h - P_c\|_2^2 + \alpha^2 \|\sigma(P_c)\|_2^2. \tag{B.5}$$

In particular, the perturbation does not introduce a bias in the target center (in expectation), and the additional term is a constant w.r.t. h.

Proof. Eq.B.4 follows from $\mathbb{E}[\xi] = 0$. For Eq.B.5, expand $\|h - P_c - \alpha\sigma(P_c) \odot \xi\|^2$; the cross term vanishes in expectation, and $\mathbb{E}\|\sigma(P_c) \odot \xi\|^2 = \|\sigma(P_c)\|^2$ since $\mathbb{E}[\xi_i^2] = 1$.

Proposition 4 (Unbiased gradient w.r.t. the feature). Let $L(h, \tilde{P}_c) = \|h - \tilde{P}_c\|_2^2$. Then

$$\mathbb{E}_\xi\big[\nabla_h L(h, \tilde{P}_c)\big] = \nabla_h \|h - P_c\|_2^2. \tag{B.6}$$

Proof. $\nabla_h \|h - \tilde{P}_c\|^2 = 2(h - \tilde{P}_c)$. Taking expectation and using Eq.B.4 yields $2(h - P_c)$.

Interpretation. Prototype perturbation preserves the optimization direction in expectation (no systematic bias), while injecting controlled stochasticity whose scale is governed by α and $\sigma(P_c)$, which empirically helps capture intra-class variability and alleviates mode collapse.

B.4 Why Feature Alignment and Class Consistency Are Complementary

We formalize a sufficient condition showing that, if the frozen classifier H_{global} has a margin at the class prototype, then generated features sufficiently close to the prototype can be correctly classified. This explains how L_{fm} supports L_{cls} during generator optimization.

Assumption 2 (Lipschitz classifier). There exists $L > 0$ such that for any h, h',

$$\|H_{\text{global}}(h) - H_{\text{global}}(h')\|_\infty \leq L\|h - h'\|_2. \tag{B.7}$$

Assumption 3 (Margin at the prototype). For class c, assume the logits satisfy

$$[H_{\text{global}}(P_c)]_c - \max_{r \neq c}[H_{\text{global}}(P_c)]_r \geq \gamma. \tag{B.8}$$

Lemma (Prototype neighborhood implies correct class prediction).

$$\|h - P_c\|_2 \leq \frac{\gamma}{2L} \implies \arg\max_r [H_{\text{global}}(h)]_r = c. \tag{B.9}$$

Proof. By Eq. B.7, each logit changes by at most $L\|h - P_c\|_2$ in absolute value. Under $\|h - P_c\|_2 \leq \gamma/(2L)$, the true-class logit decreases by at most $\gamma/2$, while any competing logit increases by at most $\gamma/2$. Therefore, the original classification margin is preserved, and the predicted class remains c.

Implication for Eq. 14. L_{fm} (Eq. 12) encourages generated features to approach the corresponding prototype neighborhood, making the sufficient condition in Eq. B.9 easier to satisfy. Meanwhile, L_{cls} (Eq. 13) directly enforces class consistency through H_{global}, which is particularly beneficial when the prototype margin is insufficient or when feature alignment alone cannot guarantee correct classification. Therefore, prototype-based feature alignment and classifier-guided class consistency provide complementary constraints rather than redundant objectives.

B.5 Stability of Stage II with the Anchor Term in Eq.(5)

Stage II introduces labeled synthetic supervision, which may cause representation drift on real samples if unconstrained. HOSFL uses the anchor term in Eq.5:

$$L_{feat} = \mathbb{E}_{x \sim D_k}\|f_k(x) - f_k^{old}(x)\|_2^2,$$

to stabilize updates on real data.

Proposition 5 (A direct bound on representation drift). Let the Stage-II objective be

$$J(\theta_k) = L_{\text{ssl2}} = L_{\text{ssl1}}(\text{real}) + \lambda_{\text{feat}} L_{\text{feat}}(\text{real}) + \lambda_{\text{ce}} L_{\text{ce}}(\text{fake}),$$

where each term is nonnegative. If during optimization the objective value does not exceed its initial value, i.e., $J(\theta_k) \leq J(\theta_k^{old})$, then

$$L_{\text{feat}}(\text{real}) \leq \frac{J(\theta_k^{old})}{\lambda_{\text{feat}}}.$$

Proof. Since all terms in $J(\theta_k)$ are nonnegative, we have

$$\lambda_{\text{feat}} L_{\text{feat}}(\text{real}) \leq J(\theta_k).$$

Under the assumption $J(\theta_k) \leq J(\theta_k^{old})$, it follows that

$$L_{\text{feat}}(\text{real}) \leq \frac{J(\theta_k^{old})}{\lambda_{\text{feat}}}.$$

This completes the proof.

Interpretation. A larger λ_{feat} enforces a smaller upper bound on real-sample representation drift, allowing the client to absorb labeled supervision from synthetic samples (Eq.6) while preserving the real-data representation structure learned in Stage I. This is particularly important under one-shot constraints, where iterative correction via repeated client-server interactions is not available.

C Additional Experiments

This appendix provides additional experimental results and implementation details to complement the main paper, with a particular focus on the behavior of the proposed method under more challenging data heterogeneity conditions. We briefly revisit the design of the two-stage client-side self-supervised training in Sec.3.1 to clarify its role under the one-shot communication constraint. Specifically, the second stage does *not* involve another round of interactive client–server optimization. Instead, the BYOL-style objective is evaluated exclusively on real data, while synthetic data are incorporated only through auxiliary stabilizing constraints, including the feature alignment loss on real samples L_{feat} in Eq. 5 and the supervised loss on synthetic samples L_{ce} in Eq. 6, which are combined in Eq. 7 to form the second-stage objective L_{ssl2}. The supplementary experiments in this appendix are therefore dedicated to evaluating the robustness of the proposed framework under extreme class heterogeneity and missing-class scenarios, as well as providing additional experimental details to facilitate reproducibility. The rest of this appendix is organized as follows. Sec. C.1 provides additional notes on the experimental setup, including evaluation protocols, implementation details, and extended descriptions of baseline methods. Sec. C.2 reports results under more extreme class heterogeneity and missing-class conditions.

C.1 Additional Notes and Baseline Details

Consistent settings and evaluation protocol. Unless otherwise specified, all supplementary experiments follow the same implementation and data partition protocol as in the main paper. All experiments were conducted on a cloud platform using NVIDIA RTX 4090 GPUs with 60GB memory, and the code is based on Python and PyTorch. We conduct experiments under the SplitFed framework with the same ResNet-18 backbone and the same split location: the client-side front model extracts intermediate representations, while the server-side tail completes the remaining computation and outputs final predictions. For fair comparisons, all methods are implemented and compared under the same ResNet-18 split architecture and a unified training protocol. In the second stage, we mix synthetic samples with real samples according to the ratio parameter ($ratio_{fake}$) and apply the additional constraints weighted by λ_{feat} and λ_{ce}. We also keep the default training schedule consistent with the main paper (e.g., generator training epochs E_G and second-stage client epochs E_{ssl2}). We report test classification accuracy under each dataset and heterogeneity level parameterized by α, consistent with the main paper.

Baselines. We compare against the following representative baselines:

FedAvgMcMahan et al. (2017). The classic federated learning approach: each client updates the full model locally and uploads model parameters to the server, which performs weighted averaging to obtain the global model. In our SplitFed setting, FedAvg serves as a reference for full-model federated training, highlighting the differences introduced by model splitting and one-shot constraints.

SL (Split Learning)Gupta & Raskar (2018). Split learning trains a partitioned model between client and server in a sequential manner: the client computes smashed features with the front model, the server computes the tail model and backpropagates gradients w.r.t. smashed features. We include SL as a "pure split training" baseline to contrast with SFL-style parallelism and communication patterns.

SFL (Split Federated Learning / SplitFed)Thapa et al. (2022). SFL integrates FL-style multi-client parallel collaboration with SL-style model partitioning: clients in parallel produce smashed features, the server completes tail-side backpropagation and returns gradients, and (parts of) the model can be federatedly aggregated across clients. Since our method targets one-shot communication in the SFL paradigm, this baseline most directly reflects the gap between conventional multi-round SFL and one-shot SFL.

MultiSFLXia et al. (2025). MultiSFL extends SFL by maintaining multiple models/branches and leveraging knowledge replay: multi-model aggregation improves robustness, while replay mitigates forgetting and enhances convergence quality. We adopt MultiSFL as a stronger SFL baseline to test whether our gains persist under a competitive split-learning variant.

FedFSA (Cross-silo Feature Space Alignment) and our SplitFed adaptation Qi et al. (2025b). FedFSA addresses cross-silo and imbalanced-client federated learning by mitigating representation

shifts through server-side feature-space alignment and calibration. Its core formulation combines clustering-based prototype and variance estimation over client features with server-side calibration updates driven by augmented representations. For a fair comparison under the SplitFed paradigm, we instantiate FedFSA in the smashed-feature space. Clients compute smashed features using the front model and participate in standard split training with the server-side tail via cross-entropy supervision. In parallel, smashed features and their labels are transmitted to a federated alignment module, which performs feature augmentation and conducts an additional calibration step on the server-side tail model or its associated alignment components. The calibrated server-side parameters are then redistributed to clients. This adaptation preserves the split-learning constraint that optimization signals predominantly flow through the smashed-feature interface, while closely aligning with FedFSA's original design of feature-space alignment and server-side calibration, thereby enabling an apples-to-apples comparison.

C.2 Supplementary Results under Extreme Class Heterogeneity

We further consider a substantially more challenging label heterogeneity setting than in the main paper. Specifically, the label space is evenly partitioned across clients such that each client observes only a subset of classes, and the class subsets are mutually disjoint. Consequently, each class is present on exactly one client, forming an extreme missing-class regime while maintaining balanced data volume across clients. Under this setting, we report classification accuracy to evaluate the robustness of different methods. To illustrate the severity of the resulting data heterogeneity, we additionally provide a visualization of the client–class distribution, as shown in Fig.8.

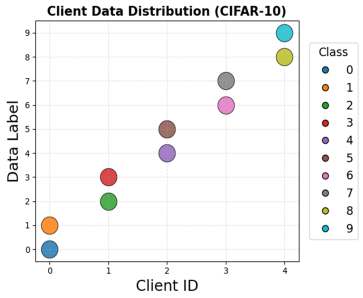

Figure 8: Client data distribution on CIFAR-10 under extremely heterogeneous settings.

Table 3: Accuracy of different methods on different datasets under extreme client data distributions

Dataset	SVHN	CIFAR10	CIFAR100
FedAvg	28.26	20.24	17.34
SL	27.74	20.77	16.41
SFL	31.03	20.17	15.23
FedFSA	40.21	27.62	19.29
HOSFL (Ours)	**46.33**	**33.25**	**27.89**

Table 3 reports the classification accuracies on SVHN, CIFAR10, and CIFAR100 under this extreme disjoint-class regime. We exclude MultiSFL from this comparison because its standard protocol is tailored to a large-client, partial-participation setting and critically relies on knowledge replay from a substantial pool of inactive clients; instantiating it with a small number of clients and mutually disjoint class supports would require non-trivial protocol modifications and thus would not constitute a directly comparable baseline.

Overall, HOSFL consistently achieves the best performance across all datasets, demonstrating strong robustness under severe label heterogeneity. Compared with traditional baselines, HOSFL achieves substantial accuracy improvements, while FedFSA provides the strongest performance among the evaluated competing methods. Notably, the performance gap between HOSFL and FedFSA becomes larger on the more challenging CIFAR-100 benchmark, where missing-class effects are more severe. This observation indicates that HOSFL can effectively mitigate the adverse effects of severe class imbalance and missing-class phenomena, maintaining stable performance even when client label distributions become highly fragmented. Consequently, HOSFL achieves consistent improvements under extreme heterogeneity, highlighting the benefits of prototype-guided generation and synthetic-data-assisted refinement in the second training stage.

C.3 Additional Performance Comparison under Different Client Scales

To further investigate the robustness of HOSFL under different federation scales, we evaluate its performance with varying numbers of clients. Specifically, we consider three client settings, including

ten clients in the main experiments, five clients, and twenty clients, under different heterogeneity levels controlled by the Dirichlet parameter α. The results are reported in Tables 1, 4, and 5.

As α decreases, client data become increasingly heterogeneous and class imbalance becomes more severe, resulting in performance degradation for all compared methods. Nevertheless, HOSFL consistently achieves strong performance across different datasets, heterogeneity levels, and client scales. This demonstrates that the proposed prototype-guided class-aware supervision mechanism can effectively utilize fragmented one-shot representations to alleviate the impact of imbalanced and incomplete class distributions.

In the ten-client setting (Table 1), we compare HOSFL with representative baselines under both their original multi-round protocols and the one-shot communication constraint. Although existing SFL approaches can achieve competitive performance through repeated client-server interactions, such iterative optimization introduces additional communication overhead and limits their applicability in communication-constrained scenarios. In contrast, HOSFL completes training with only a single communication round while maintaining competitive or superior performance compared with multi-round baselines in various settings. Moreover, when these methods are directly restricted to one-shot communication, their performance decreases substantially, highlighting the challenge of adapting iterative approaches to the one-shot split federated learning setting. These results demonstrate that HOSFL can construct effective class-aware supervision from limited one-shot uploaded representations.

The five-client and twenty-client experiments further evaluate the robustness of HOSFL under different federation scales. In the five-client setting (Table 4), HOSFL achieves consistent improvements over compared methods. Particularly, on CIFAR-100 with $\alpha = 0.1$, HOSFL obtains 33.10% accuracy, outperforming the strongest baseline MultiSFL by 3.99%. Similar trends are observed in the twenty-client setting (Table 5). For example, on CIFAR-10 with $\alpha = 0.1$, HOSFL achieves 48.68% accuracy, exceeding MultiSFL, FedFSA, and SFL by 2.28%, 5.01%, and 7.33%, respectively.

Overall, the consistent improvements across different datasets, heterogeneity levels, and federation scales demonstrate that HOSFL remains effective when the number of participating clients varies. These results validate the robustness of HOSFL and its capability to alleviate fragmented class distributions through prototype-guided class-aware generation under communication-constrained split federated learning.

Table 4: Test accuracy (%) comparison of different methods under different heterogeneity levels α with 5 clients on three benchmark datasets.

Dataset	Hetero. Settings	Accuracy (%)					
		FedAvg	SL	SFL	MultiSFL	FedFSA	HOSFL (Ours)
SVHN	$\alpha = 0.1$	65.45	61.58	59.44	68.53	69.36	**75.22**
	$\alpha = 0.3$	77.65	70.47	69.01	79.83	87.45	**88.81**
	$\alpha = 0.5$	84.74	80.42	84.39	83.94	88.41	**89.54**
CIFAR-10	$\alpha = 0.1$	39.94	37.87	38.61	44.70	41.18	**45.84**
	$\alpha = 0.3$	54.04	51.68	54.94	54.74	58.71	**60.42**
	$\alpha = 0.5$	59.76	58.31	64.87	56.79	63.23	**66.31**
CIFAR-100	$\alpha = 0.1$	26.42	24.22	25.46	28.80	29.02	**33.10**
	$\alpha = 0.3$	27.85	27.59	27.18	30.93	30.36	**35.18**
	$\alpha = 0.5$	29.78	29.43	30.68	32.64	33.95	**36.45**

C.4 Client Resource Overhead Analysis

To investigate whether HOSFL introduces excessive computational and memory burdens on clients, we further evaluate the resource consumption during local training. Specifically, we record the training time of the two client-side optimization stages, peak GPU memory usage, and the number of parameters stored on the client side. The results for ten clients are reported in Table 6.

As shown in Table 6, the computational cost varies across clients mainly due to the difference in local sample sizes. The first and second training stages require approximately 100–347 seconds and 64–237 seconds, respectively, where clients with more local samples generally incur higher

Table 5: Test accuracy (%) comparison of different methods under different heterogeneity levels α with 20 clients on three benchmark datasets.

Dataset	Hetero. Settings	Accuracy (%)					
		FedAvg	SL	SFL	MultiSFL	FedFSA	HOSFL (Ours)
SVHN	$\alpha = 0.1$	74.63	71.78	63.68	87.17	80.74	**87.81**
	$\alpha = 0.3$	78.94	76.04	68.64	88.63	82.43	**91.60**
	$\alpha = 0.5$	79.65	78.83	72.75	89.71	84.96	**92.41**
CIFAR-10	$\alpha = 0.1$	40.64	37.34	41.35	46.40	43.67	**48.68**
	$\alpha = 0.3$	52.10	49.68	57.68	59.56	60.64	**62.01**
	$\alpha = 0.5$	54.63	52.74	65.69	68.73	64.93	**69.12**
CIFAR-100	$\alpha = 0.1$	25.80	24.73	28.80	32.34	28.76	**34.31**
	$\alpha = 0.3$	30.21	29.69	31.96	35.68	32.69	**38.13**
	$\alpha = 0.5$	33.68	32.93	33.74	38.68	33.70	**39.49**

training time. This indicates that the additional client computation introduced by HOSFL is mainly determined by local data processing rather than extra model complexity.

Table 6: Client resource overhead of HOSFL.

Client	Samples	Stage-I Time(s)	Stage-II Time(s)	Memory(MB)	Front Params
0	6826	346.59	236.90	1242.20	297728
1	6985	345.83	228.46	1194.99	297728
2	3816	214.13	140.59	1297.67	297728
3	1765	100.37	64.22	1299.96	297728
4	3237	182.61	118.57	1302.25	297728
5	2291	127.63	82.62	1304.54	297728
6	6356	329.17	213.06	1257.58	297728
7	6346	331.71	220.37	1259.87	297728
8	6261	326.67	219.63	1260.41	297728
9	6117	325.75	201.87	1263.84	297728

Meanwhile, the peak GPU memory consumption remains within 1.20–1.31 GB across all clients, demonstrating stable memory requirements under heterogeneous data distributions. Moreover, each client only stores the front model containing 297,728 parameters, which is substantially smaller than maintaining the complete backbone model. These results verify that HOSFL does not impose excessive resource requirements on clients and remains applicable to resource-constrained devices.

C.5 Detailed Privacy Analysis

To provide a comprehensive analysis of the privacy behavior of HOSFL, we evaluate both intermediate feature leakage and synthetic data privacy under the one-shot split federated learning setting.

Experimental Setup. The privacy evaluation is conducted on CIFAR-10 with heterogeneous client data distributions generated by the Dirichlet partition ($\alpha = 0.3$). We compare HOSFL with conventional SFL under the same split architecture and feature transmission protocol. Following the privacy evaluation setting in SFL, both methods apply the same differential privacy mechanism during intermediate feature transmission, including identical noise perturbation strategies. Therefore, the comparison mainly reflects the influence of the communication paradigm rather than differences in privacy-preserving mechanisms.

Intermediate Feature Leakage Analysis. Fig. 6 reports the feature reconstruction quality measured by Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). These metrics quantify the similarity between reconstructed samples and the corresponding original training images. Higher PSNR and SSIM values indicate that intermediate representations contain more recoverable visual information and imply higher empirical leakage risks.

As shown in Fig. 6, HOSFL consistently achieves lower PSNR and SSIM values than conventional SFL across different clients. Specifically, HOSFL obtains PSNR values around 10–10.6 dB and SSIM values around 0.02–0.04, indicating that the transmitted intermediate representations are more difficult to reconstruct into visually similar samples. This reduction mainly benefits from the one-shot

communication design, which avoids repeated feature exchanges between clients and servers during iterative optimization and reduces potential feature exposure opportunities.

Synthetic Data Privacy Analysis. Beyond intermediate feature transmission, we further investigate whether the synthetic samples generated by the federated server reveal identifiable information from client data. Fig. 7 compares real CIFAR-10 training images with synthetic samples generated by HOSFL. The first two rows present real images, while the last two rows show generated samples.

As illustrated in Fig. 7, the generated samples do not preserve recognizable object contours, fine-grained textures, or pixel-level patterns from the original images. Instead, they exhibit substantial visual differences from real training samples, suggesting that the generator does not memorize or directly reconstruct client data during synthesis. Therefore, the generated samples provide auxiliary training information without apparent direct disclosure of original data.

Meanwhile, the synthetic samples are not equivalent to random noise. Although they lack visually interpretable structures, they preserve class-discriminative information in the feature space through prototype-guided feature alignment and classifier-guided class consistency. Such information enables the generated samples to effectively support second-stage client-side refinement.

Overall, these results demonstrate that HOSFL reduces empirical privacy leakage risks from both intermediate feature transmission and synthetic sample generation perspectives, while maintaining effective knowledge transfer under the one-shot communication constraint.