LABEL DISTRIBUTION INFERENCE ATTACK AND DIRECTIONAL DEFENSE ON PROTOTYPES IN FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Prototype-based federated learning has emerged as an effective approach for addressing model heterogeneity by enabling clients with diverse architectures to collaborate through the exchange of per-class mean feature embeddings. Although prototypes are generally considered privacy-preserving, we show that they can reveal sensitive information about clients' label distributions. Specifically, we identify a geometric vulnerability: local prototypes exhibit stronger angular alignment with global prototypes for classes with more local samples, making prototype alignment an informative indicator of class frequency. Motivated by this observation, we propose the *Prototype-Based Label Distribution Inference Attack* (PLIA), which infers per-client label distributions solely from shared prototypes by exploiting this geometric property. To mitigate this privacy risk, we further introduce *Directional Prototype Perturbation* (DPP), a defense mechanism that applies orthogonal perturbations with randomized per-class budgets to disrupt this leakage signal. Experiments across five benchmark datasets show that PLIA outperforms or matches baseline attacks, while DPP effectively reduces the label-distribution leakage PLIA exploits with only a marginal impact on model performance.

1 INTRODUCTION

Preserving data privacy while enabling collaborative training across distributed clients is a central challenge in modern machine learning (McMahan et al., 2017). Federated learning (FL) has addressed this challenge with notable success in sensitive domains such as healthcare and finance (Zhang et al., 2021; Kairouz et al., 2021; Pati et al., 2024). As a foundational FL approach, FedAvg (McMahan et al., 2017) aggregates full model parameters from clients on a central server, forming a single global model that is then distributed back to clients for further local training. However, this conventional parameter-sharing paradigm rests on a critical assumption: that all clients share homogeneous model architectures. In practice, clients often differ substantially in computational capacity, memory, and hardware — making model heterogeneity a fundamental and pervasive challenge (Fan et al., 2024).

This architectural constraint has motivated model-heterogeneous FL, a new paradigm that enables clients to use different model architectures (Fan et al., 2024). Existing approaches fall into two broad categories: partial training (PT)-based methods, which allow clients to train specific segments of a neural network according to their computational capabilities (Diao et al., 2020; Horvath et al., 2021; Zhu et al., 2022; Alam et al., 2022), and knowledge distillation (KD)-based methods, which offer an architecture-free alternative by transferring knowledge through such as logits (Jeong et al., 2018; Li & Wang, 2019) or intermediate features (Tan et al., 2022a; Zhang et al., 2024a). Among KD-based methods, prototype-based FL (PBFL) has become particularly practical, sharing per-class mean activations from the penultimate layer as compact, communication-efficient representations (Tan et al., 2022a). Beyond efficiency, prototypes are widely regarded as privacy-preserving alternatives to sharing raw parameters (Tan et al., 2022a; Zhang et al., 2024a).

Despite their privacy advantage over direct model sharing, transmitting prototypes still exposes clients to non-trivial privacy risks (Wang et al., 2025; 2026). Prototypes aggregate feature information into class-level representations and can retain traces of individual samples—particularly with small local datasets—leaving them vulnerable to membership inference and reconstruction attacks (Wang et al.,

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.

2026). While prior work has studied sample-level prototype privacy through differential privacy (Tan et al., 2022b; Wang et al., 2024; Zhang et al., 2024b; Wang et al., 2026), label distribution leakage from prototypes remains largely unexamined — despite directly revealing the aggregate composition of a client's local data, a dimension of privacy orthogonal to sample-level threats.

In this work, we focus on *label distribution inference attacks*, which aim to recover per-class sample proportions from a client's local dataset. Although a variety of such attacks exist (Zhou et al., 2022; Gu & Bai, 2023; Jiang et al., 2024b;a; Dai et al., 2024), they typically target model parameter sharing and often require auxiliary or synthetic datasets, or even meta-classifiers, making them inapplicable to the prototype-sharing setting. The vulnerability of prototypes to label distribution inference thus remains an open problem.

Our key observation is that prototype regularization used in PBFL pulls local prototypes toward global prototypes more strongly for classes with more local samples. Hence, their angular alignment carries information about the client's label distribution. Building on this insight, we study label distribution leakage in PBFL from both the attack and defense perspectives. On the attack side, we introduce the *Prototype-Based Label Distribution Inference Attack* (PLIA), which exploits this angular alignment to infer each client's label distribution. On the defense side, we propose *Directional Prototype Perturbation* (DPP), which perturbs each local prototype along the component of the global prototype orthogonal to the local prototype, with class-wise randomized budgets that further break the relative label ranking while minimizing distortion to prototype utility. Our contributions are as follows:

- We identify label distribution inference as a novel privacy threat in PBFL: beyond the already exposed class presence of each client, an honest-but-curious server can infer client label distributions solely from shared prototypes.
- We propose both attack and defense methods: PLIA, which recovers client label distributions via angular alignment between local and global prototypes, and DPP, a principled defense that applies orthogonal, class-wise randomized perturbations to disrupt the angular signal.
- Through extensive evaluation across five datasets and diverse FL settings, we demonstrate that PLIA is competitive with baseline attacks that require access to model updates, and that DPP effectively mitigates leakage while preserving model performance.

2 Related Work

Label Distribution Leakage in FL. Although FL approaches do not transmit clients' private raw data, label distribution information can still be leaked, as model parameters naturally encode data characteristics (Kang et al., 2020; Khakzar et al., 2021; Lee & Choi, 2025). Several attacks have been proposed to infer label distributions from clients' private data in model-sharing FL: PPA (Zhou et al., 2022) exploits gradient sensitivity to class proportions, LDIA (Gu & Bai, 2023) characterizes output layer updates via a learned network, PGDA (Jiang et al., 2024b) applies projected gradient descent to recover both feature and label distributions, LDPM (Jiang et al., 2024a) provides formal privacy guarantees via SGD perturbation, and Decaf (Dai et al., 2024) analyzes local gradient magnitudes to decompose data distributions. However, all of these attacks assume model sharing—targeting gradient or parameter updates, as in FedAvg—and thus do not apply to prototype-based FL, where no model parameters are exchanged. To our knowledge, this work is the first to study label distribution leakage from shared prototypes, from both attack and defense perspectives.

Prototype-Based FL and its Privacy. PBFL coordinates feature representations across clients by exchanging class prototypes. FedProto (Tan et al., 2022a) establishes the foundational framework by distilling knowledge from aggregated global prototypes, while FedDistill (Jeong et al., 2018) shares averaged class-wise logits as a lightweight alternative. Subsequent work has further advanced prototype-based federation through contrastive learning (Zhang et al., 2024a), geometric optimization (Choi et al., 2026), fixed semantic anchors (Zhou et al., 2025), and dual-level clustering (Wang et al., 2024). Despite growing interest in privacy-preserving prototype release, existing defenses (Tan et al., 2022b; Wang et al., 2024; Zhang et al., 2024b; Wang et al., 2026) adopt a differential privacy perspective by injecting noise into prototypes without explicitly targeting label distribution leakage. In contrast, DPP uses directional perturbations designed to disrupt the angular signal that leaks label-distribution information while preserving prototype utility.

3 BACKGROUND

This section formalizes the heterogeneous FL setting, introduces prototype-based knowledge sharing, and presents an empirical analysis motivating our approach.

3.1 FEDERATED LEARNING FRAMEWORK

We consider a heterogeneous federated learning setting for classification tasks, where data distributions and model architectures vary across clients. Formally, the environment consists of a central server connected to M clients, each utilizing a distinct model architecture and maintaining its private data distribution P_i over labeled samples $(x, y) \in \mathcal{X} \times \{1, \dots, K\}$. Following FedProto (Tan et al., 2022a), we partition each client's network $\boldsymbol{w}_i$ into a feature extractor f_i with parameters $\boldsymbol{\theta}_i$ that maps inputs into a d-dimensional feature space, and a classifier g_i with parameters $\boldsymbol{\phi}_i$ that maps features to pre-softmax logits in $\mathbb{R}^K$. For a sample x, the feature vector is $\boldsymbol{c}_i(x) = f_i(\boldsymbol{\theta}_i; x) \in \mathbb{R}^d$ and the logit vector is $\boldsymbol{z}_i(x) = g_i(\boldsymbol{\phi}_i;\, \boldsymbol{c}_i(x)) \in \mathbb{R}^K$. The federated learning objective is to minimize the average expected loss across all clients:

$$\min_{\{\boldsymbol{\theta}_i, \boldsymbol{\phi}_i\}_{i=1}^M} \frac{1}{M} \sum_{i=1}^{M} \mathbb{E}_{(x,y)\sim P_i} \left[-\log \frac{\exp(z_i^{(y)}(x; \boldsymbol{\theta}_i, \boldsymbol{\phi}_i))}{\sum_{k=1}^{K} \exp(z_i^{(k)}(x; \boldsymbol{\theta}_i, \boldsymbol{\phi}_i))} \right]. \tag{1}$$

3.2 PROTOTYPES AS SHARED KNOWLEDGE IN FEDERATED LEARNING

Heterogeneous model architectures preclude direct parameter sharing, motivating the use of *prototypes* as an architecture-agnostic medium for knowledge exchange. While the term *prototype* is used inconsistently in the FL literature—commonly referring to mean feature embeddings from the penultimate layer (Tan et al., 2022a; Zhang et al., 2024a; Lee & Choi, 2026) but sometimes to classifier head parameters (Dai et al., 2023)—we adopt the former convention and extend it to two complementary types: *feature prototypes* and *logit prototypes*. Both types are exchanged between clients and the server following three common steps, each described below using feature prototypes as the primary exposition.

Local Feature Prototype Generation. A *local feature prototype* of class j at client i is the mean feature vector over all local samples of class j:

$$\bar{\boldsymbol{c}}_{i,j}^{L} = \frac{1}{n_{i,j}} \sum_{(x,y)\in\mathcal{D}_{i,j}} \boldsymbol{c}_i(x), \tag{2}$$

where $n_{i,j} = |\mathcal{D}_{i,j}|$ and $\mathcal{D}_{i,j} \subseteq \mathcal{D}_i$ is the local dataset of client i restricted to class j. Each client uploads its local feature prototype set $\{\bar{\boldsymbol{c}}_{i,j}^{L}\}_{j\in\mathcal{Y}_i}$ to the server, where $\mathcal{Y}_i$ denotes the set of classes present in client i's local data.

Global Feature Prototype Generation. The server aggregates the received local feature prototypes into a *global feature prototype* via simple averaging:

$$\bar{\boldsymbol{c}}_{j}^{G} = \frac{1}{|\mathcal{N}_j|} \sum_{i\in\mathcal{N}_j} \bar{\boldsymbol{c}}_{i,j}^{L}, \tag{3}$$

where $\mathcal{N}_j$ denotes the set of clients holding class j. Although sample-weighted averaging is theoretically more principled, it requires clients to disclose their class-wise sample counts, which are themselves sensitive information. Simple averaging is therefore widely adopted in practice (Tan et al., 2022a; Zhang et al., 2024a), and we follow this convention throughout. The global feature prototypes $\{\bar{\boldsymbol{c}}_{j}^{G}\}_{j=1}^{K}$ are then broadcast back to all clients to guide local training.

Local Training via Feature Prototype Distillation. Upon receiving the global feature prototypes, each client minimizes a combined loss:

$$\tilde{\mathcal{L}}_i(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i) = \mathcal{L}_i(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i) + \lambda \mathcal{R}_i, \tag{4}$$

where λ controls the regularization strength and $\mathcal{R}_i$ is the prototype distillation regularizer. When feature prototypes are shared, $\mathcal{R}_i = \mathcal{R}_i^F$, which penalizes the distance between local and global feature prototypes:

$$\mathcal{R}_i^F = \sum_{j\in\mathcal{Y}_i} \rho\big(\bar{\boldsymbol{c}}_{j}^{G},\, \bar{\boldsymbol{c}}_{i,j}^{L}\big)\,, \tag{5}$$

where $\rho(\cdot,\cdot)$ denotes the squared Euclidean distance. In implementation, this regularization is applied at the sample level. For each local sample (x, y), its feature representation $\boldsymbol{c}_i(x)$ is encouraged to approach the corresponding global prototype $\bar{\boldsymbol{c}}_y^G$. Consequently, classes with more local samples contribute more frequently to the regularization loss during local optimization.

Extension to Logit Prototypes. Logits are the pre-softmax outputs of the classifier head and are widely used in knowledge distillation as compact class predictions, making them a natural candidate for prototype sharing. The *logit prototype* follows the same pipeline as the feature prototype, with two differences. First, the feature vector $\boldsymbol{c}_i(x)$ is replaced by the logit vector $\boldsymbol{z}_i(x) \in \mathbb{R}^K$, yielding the *local logit prototype*:

$$\bar{\boldsymbol{z}}_{i,j}^L = \frac{1}{n_{i,j}} \sum_{(x,y)\in\mathcal{D}_{i,j}} \boldsymbol{z}_i(x). \tag{6}$$

The server aggregates these identically to Eq. (3), producing the *global logit prototype* $\bar{\boldsymbol{z}}_j^G$. Second, the squared Euclidean distance in $\mathcal{R}_i^F$ is replaced by cross-entropy between the softmax-induced distributions, since logit vectors encode class probabilities up to a softmax transformation (Jeong et al., 2018):

$$\mathcal{R}_i^Z = \sum_{j\in\mathcal{Y}_i} \varphi\big(\mathrm{softmax}(\bar{\boldsymbol{z}}_j^G),\, \mathrm{softmax}(\bar{\boldsymbol{z}}_{i,j}^L)\big)\,, \tag{7}$$

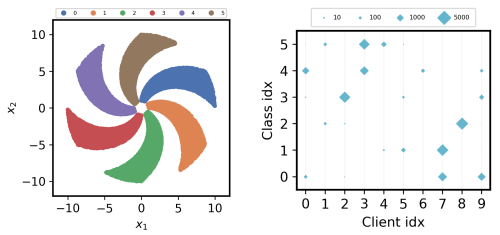
where $\varphi(\cdot,\cdot)$ denotes the cross-entropy.

3.3 Motivating Analysis

From Eqs. (5) and (7), the local prototypes are pulled toward the global prototypes during local training. This raises a natural question: *what information is encoded in the geometric relationship between local and global prototypes?* Because the regularizations are applied to each local sample and the local prototype is the mean feature over these samples (Eq. (2)), classes with more samples receive stronger cumulative gradient pressure toward their global prototypes (see Appendix A for the underlying intuition), which tends to yield stronger angular alignment. Consequently, the angular distance between local and global prototypes can serve as a proxy for relative class frequency within a client—classes that make up a larger proportion of a client's data exhibit smaller angular deviation. The same reasoning applies to logit prototypes.

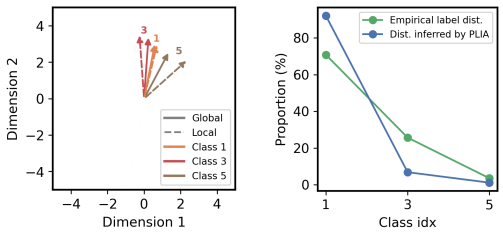
Empirical Illustration. We illustrate this intuition using a synthetic spiral dataset in two dimensions (Figure 1a) (Dai et al., 2023), comprising six classes with 5000 points each (generation details in the Appendix B). We train a 5-layer MLP with ReLU activations in a two-dimensional feature space for visualization, across 10 clients in a non-IID setting, where data is partitioned via a Dirichlet distribution with $\alpha = 0.1$ (Figure 1b). We train FedProto for 200 FL rounds and visualize the global and local feature prototypes at the final round for client #5, which holds only classes #1, #3, and #5. As shown in Figure 2a, class #1, which occupies the largest local proportion, has its local prototype closely aligned with the global prototype. In contrast, class #5, which is underrepresented, shows a large angular deviation from its global counterpart—visually confirming our intuition.

We further validate this by computing $1/(1-\mathrm{sim}(\bar{\boldsymbol{c}}_j^G, \bar{\boldsymbol{c}}_{i,j}^L))$ for each class, where $\mathrm{sim}(\boldsymbol{a},\boldsymbol{b}) = \frac{\boldsymbol{a}\cdot\boldsymbol{b}}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$ denotes the cosine similarity, and comparing it against the true sample proportions (Figure 2b). The inferred distribution preserves the correct class ranking, showing that the angular geometry of prototypes carries meaningful information about local label distributions—and highlighting a potential inference attack on the server side under an appropriate threat model.



(a) Spiral dataset (b) Client data distribution

Figure 1: Spiral dataset with six classes and its non-IID distribution across 10 clients.



(a) Prototype geometry (b) Inferred distribution

Figure 2: Prototype geometry and PLIA-inferred vs. empirical label distributions.

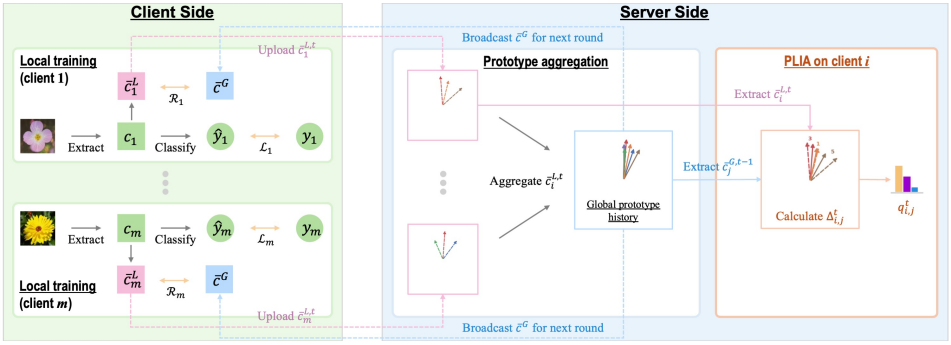


Figure 3: Overview of the FedProto protocol and the PLIA mechanism.

4 METHODOLOGY

This section formalizes the threat model and then presents our proposed attack and defense methods.

4.1 THREAT MODEL AND PRIVACY METRICS

Threat Model. We consider an honest-but-curious server that faithfully follows the federated learning protocol while passively attempting to infer clients' private label distributions. At each communication round t, the server observes the local prototypes uploaded by individual clients—either feature prototypes $\bar{\boldsymbol{c}}_{i,j}^{L,t}$ or logit prototypes $\bar{\boldsymbol{z}}_{i,j}^{L,t}$—and has access to the global prototypes $\bar{\boldsymbol{c}}_j^{G,t-1}$ or $\bar{\boldsymbol{z}}_j^{G,t-1}$ broadcast at round $t-1$. Following prior inference attacks (Zhou et al., 2022; Dai et al., 2024), we assume that prototypes are uploaded in plaintext without secure aggregation (Liu et al., 2022), which is incompatible with methods such as FedTGP that require individual client prototypes. Secure-aggregation settings are beyond the scope of this work.

Privacy Metrics. We quantify *Label Distribution Leakage* (LDL) by comparing each client's empirical label distribution $\hat{\mathbf{p}}_i = (\hat{p}_{i,1}, \ldots, \hat{p}_{i,K})$, where $\hat{p}_{i,j} = n_{i,j}/N_i$ and $N_i = \sum_{k=1}^{K} n_{i,k}$, with the adversary's estimate $\mathbf{q}_i$, using total variation distance ($\mathrm{LDL}_i^{\mathrm{TV}}$), Kullback–Leibler divergence ($\mathrm{LDL}_i^{\mathrm{KL}}$), and Spearman rank correlation ($\mathrm{LDL}_i^{\mathrm{SC}}$). Greater leakage corresponds to lower TV and KL and higher SC. We report the average over all M clients, $\overline{\mathrm{LDL}}^{s} = \frac{1}{M}\sum_{i=1}^{M} \mathrm{LDL}_i^{s}$ for $s \in \{\mathrm{TV}, \mathrm{KL}, \mathrm{SC}\}$, with formal definitions in Appendix C.

4.2 ATTACK: PROTOTYPE-BASED LABEL DISTRIBUTION INFERENCE ATTACK (PLIA)

As shown in Section 3.3, classes with more local samples receive stronger cumulative gradient pressure through the regularization in Eqs. (5) and (7), pulling their local prototypes closer to the global prototypes. This pull also tends to reduce the angle between them, which is insensitive to prototype magnitude. PLIA exploits this signal: since the server observes both global and local prototypes at each round, it measures their angular alignment via cosine similarity to score class prevalence (Figure 3). For client i and class j at round t, the server computes the alignment score:

$$\Delta_{i,j}^{t} = \frac{1}{1 - \mathrm{sim}\left(\bar{\boldsymbol{c}}_j^{G,t-1},\, \bar{\boldsymbol{c}}_{i,j}^{L,t}\right)}, \tag{8}$$

where $\mathrm{sim}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}\cdot\boldsymbol{b}}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$. In practice, we observe $\mathrm{sim}(\cdot,\cdot) \geq 0$, so the cosine distance $1 - \mathrm{sim}(\cdot,\cdot)$ lies in $[0, 1]$. Thus, $\Delta_{i,j}^{t}$ grows as the local prototype aligns more closely with the global one, indicating a larger proportion of class j in client i's data.

Since clients upload prototypes only for the classes they hold, the server can identify the set of classes held by client i, denoted by $\mathcal{Y}_i$. PLIA therefore aims to infer the *proportions* among these classes. Normalizing alignment score $\Delta_{i,j}^{t}$ over $\mathcal{Y}_i$ gives the inferred label distribution $\mathbf{q}_i^{t} = (q_{i,1}^{t}, \ldots, q_{i,K}^{t})$:

$$q_{i,j}^{t} = \begin{cases} \dfrac{\Delta_{i,j}^{t}}{\sum_{k\in\mathcal{Y}_i} \Delta_{i,k}^{t}}, & j \in \mathcal{Y}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Other Design Choices. Since the prototype regularization in Eq. (5) is based on Euclidean distance, one might expect Euclidean distance to be a more natural choice than cosine similarity for scoring class prevalence. We therefore also evaluate an alternative scoring rule based on inverse Euclidean distance between local and global prototypes, referred to as P-InvDist (Appendix G). As shown in Section 5, this alternative performs worse than PLIA, and we discuss possible reasons there.

4.3 Defense: Directional Prototype Perturbation (DPP)

DPP mitigates label-distribution leakage by perturbing local prototypes before uploading them to the server, disrupting the angular signal PLIA exploits. For each class j, let $\boldsymbol{u} = \bar{\boldsymbol{c}}_{i,j}^{L,t}$ and $\boldsymbol{v} = \bar{\boldsymbol{c}}_{j}^{G,t-1}$ denote the local and global prototype, respectively. The perturbed prototype by DPP is defined as:

$$\hat{\boldsymbol{u}} = \boldsymbol{u} + s_j \epsilon_j \cdot \frac{\boldsymbol{v}_\perp}{\|\boldsymbol{v}_\perp\|}, \tag{10}$$

where $\boldsymbol{v}_\perp$ is the perturbation direction, $s_j \sim \mathcal{U}(\{-1, +1\})$ is a random sign, and $\epsilon_j > 0$ is the per-class budget. DPP determines these components via three design choices described below.

Orthogonal Direction Construction. Since PLIA exploits the angular alignment between $\boldsymbol{u}$ and $\boldsymbol{v}$, DPP sets the perturbation direction to the component of $\boldsymbol{v}$ perpendicular to $\boldsymbol{u}$:

$$\boldsymbol{v}_\perp = \boldsymbol{v} - \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\|^2} \boldsymbol{u}. \tag{11}$$

The following proposition shows that, to first order in $\frac{\epsilon}{\|\boldsymbol{u}\|}$, perturbing along $\boldsymbol{v}_\perp$ maximizes the angular distance between the local and global prototypes under a perturbation budget.

Proposition 1 (First-Order Optimality of Perpendicular Perturbation)**.** *Let $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ with $\boldsymbol{u} \neq \boldsymbol{0}$ and $\boldsymbol{v} \nparallel \boldsymbol{u}$, and let $\epsilon > 0$ satisfy $\epsilon \ll \|\boldsymbol{u}\|$. Among all perturbations $\boldsymbol{\delta}$ with $\|\boldsymbol{\delta}\| \leq \epsilon$, the perturbation that maximizes the angular distance between $\boldsymbol{u}' = \boldsymbol{u} + \boldsymbol{\delta}$ and $\boldsymbol{v}$ to first order in $\frac{\epsilon}{\|\boldsymbol{u}\|}$ is $\boldsymbol{\delta}^* = -\epsilon \frac{\boldsymbol{v}_\perp}{\|\boldsymbol{v}_\perp\|}$, where $\boldsymbol{v}_\perp = \boldsymbol{v} - \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\|^2} \boldsymbol{u}$ is the component of $\boldsymbol{v}$ orthogonal to $\boldsymbol{u}$.*

The proof is provided in Appendix D.

Random Sign Selection. By Proposition 1, $\pm \frac{\boldsymbol{v}_\perp}{\|\boldsymbol{v}_\perp\|}$ are the directions along which the cosine similarity changes most to first order, increasing and decreasing it, respectively. Since PLIA relies on relative alignment scores across classes, perturbing all classes in the same direction would largely preserve their ranking. DPP therefore samples the sign s_j independently per class for each client.

Class-Wise Budget Allocation. Applying a uniform budget shifts every prototype nearly equally, leaving the inferred $\mathbf{q}_i$ largely unchanged. DPP instead samples independent per-class budgets $r_j \sim \mathcal{U}(0, 1)$ and normalizes: $\epsilon_j = \epsilon_{\text{total}} \frac{r_j}{\sum_{k \in \mathcal{Y}_i} r_k}$, where $\mathcal{Y}_i$ is the set of classes at client i and ϵ_{total} is the total budget, introducing asymmetric angular distortion that disrupts the inferred label ranking.

The algorithms for PLIA and DPP are provided in Appendix E (Algorithm 1).

Relationship with Differential Privacy. DPP and differential privacy (DP) (Dwork et al., 2006) serve different purposes. DP provides a formal, attack-agnostic guarantee that bounds the influence of any individual training sample on the released output, typically by adding isotropic noise calibrated to the output's sensitivity. DPP, in contrast, targets a specific distribution-level statistic, the client's label distribution, and perturbs prototypes along the direction that disrupts the angular signal exploited by PLIA. Since the two methods protect different quantities under different threat models, a direct comparison is less meaningful. Nevertheless, in Section 5, we evaluate whether PLIA remains effective under DP and compare DP with DPP as defenses against PLIA attacks.

5 Experiments

We present our experimental setup and results, and the code is in the supplementary materials.

5.1 Experimental Setup

Datasets and Models. We evaluate PLIA and DPP on five datasets spanning time-series, text, and image modalities: HAR (Anguita et al., 2013), Amazon Review (Amazon) (McAuley & Leskovec,

2013), CIFAR-10/100 (Krizhevsky et al., 2009), and Tiny ImageNet (Tiny IN) (Le & Yang, 2015). In the homogeneous setting, we use HARCNN, AmazonMLP, a 4-layer CNN, and ResNet-18 (He et al., 2016) for HAR, Amazon Review, CIFAR-10/100, and Tiny IN, respectively, with a default prototype dimension of $d = 512$. In the heterogeneous setting, following HtFLlib (Zhang et al., 2025), each client is assigned one architecture from a pool: HARCNN and AmazonMLP variants with different depths or widths for HAR and Amazon Review, and a 4-layer CNN, GoogLeNet (Szegedy et al., 2015), MobileNetV2 (Sandler et al., 2018), or ResNet-18 for the image datasets.

Baselines and Evaluation Metrics. For attack evaluation, we compare PLIA against Random, which samples label proportions uniformly and normalizes them (Dai et al., 2024); Uniform, which assigns equal proportions to all held classes; Decaf and PGDA, which infer label distributions from shared model updates in FedAvg; and P-InvDist, which replaces cosine similarity in PLIA with inverse Euclidean distance. For defense evaluation, we compare DPP against two DP-based defenses, IGPP and VPDR (Wang et al., 2026), and three ablations (RD-UB, RD-CB, and OD-UB) that isolate the effects of orthogonal direction and class-wise budget allocation. By default, DPP uses a total budget equal to the number of classes, with per-class budgets drawn from $\mathcal{U}(0, 1)$ and normalized. For each client, we randomly sample per-class signs and budgets once in the first round and fix them thereafter. We report $\overline{\text{LDL}}$, averaged over all rounds, and the best mean client accuracy as the utility metric. All results are averaged over three random seeds.

FL Configurations. Experiments use 20 clients with full participation and run for 200 communication rounds, with one local epoch per round. Client data are partitioned into a 75:25 train/test split, and heterogeneity is controlled via a Dirichlet distribution with concentration parameter $\alpha = 0.5$. We evaluate five FL methods: FedAvg for the model-sharing attacks Decaf and PGDA, and FedProto, FedTGP, FedSA, and FedDistill for the prototype-based attacks P-InvDist and PLIA.

5.2 Attack Performance and Analysis

Overall Performance. Table 1 reports PLIA's inference performance. In the homogeneous architecture setting (Homo.), PLIA outperforms the Random and Uniform baselines in all cases and achieves the lowest $\overline{\text{LDL}}^{\text{TV}}$ on four of the five datasets. PGDA attains the lowest $\overline{\text{LDL}}^{\text{TV}}$ on Tiny ImageNet and the highest $\overline{\text{LDL}}^{\text{SC}}$ on HAR, CIFAR-10, and CIFAR-100. However, this comparison should be interpreted with caution, as PGDA operates under FedAvg, where the server observes full model updates, whereas PLIA observes only class-wise prototypes. PLIA also outperforms P-InvDist in most settings, possibly because cosine similarity ignores prototype magnitude (Appendix G). PLIA

Table 1: PLIA attack performance on various datasets (TV: $\overline{\text{LDL}}^{\text{TV}}$, SC: $\overline{\text{LDL}}^{\text{SC}}$; Homo./Hete.: homogeneous/heterogeneous architectures). **Bold**: best per column. Random's SC ($\sim 10^{-4}$) is shown as 0.000; SC is undefined for Uniform.

Attack	Algo.	HAR		Amazon		CIFAR-10		CIFAR-100		Tiny IN	
		TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑
Random	–	0.467	0.000	0.268	0.000	0.485	0.000	0.481	0.000	0.485	0.000
Uniform	–	0.433	–	0.220	–	0.436	–	0.431	–	0.435	–
Decaf (Homo.)	FedAvg	0.356	0.507	0.204	0.157	0.412	0.289	0.460	0.081	0.470	0.055
PGDA (Homo.)	FedAvg	0.358	**0.809**	0.189	0.372	0.310	**0.888**	0.336	**0.948**	**0.174**	0.930
P-InvDist (Homo.)	FedProto	0.412	0.181	0.161	0.637	0.364	0.718	0.338	0.851	0.315	0.940
	FedTGP	0.273	0.679	0.145	0.583	0.249	0.837	0.373	0.689	0.325	0.878
	FedSA	0.421	0.224	0.131	0.817	0.405	0.404	0.373	0.714	0.384	0.632
	FedDistill	0.410	0.348	0.161	0.534	0.444	0.194	0.363	0.762	0.374	0.671
PLIA (Homo.)	FedProto	0.284	0.706	**0.068**	0.943	0.273	0.816	0.287	0.889	0.208	**0.947**
	FedTGP	0.280	0.648	0.111	0.555	**0.179**	0.833	0.322	0.697	0.240	0.865
	FedSA	**0.238**	0.767	0.069	**0.986**	0.258	0.775	0.283	0.890	0.357	0.710
	FedDistill	0.344	0.723	0.155	0.634	0.332	0.630	**0.256**	0.845	0.306	0.765
PLIA (Hete.)	FedProto	0.407	0.331	0.168	0.521	0.247	0.839	0.327	0.740	0.338	0.756
	FedTGP	0.301	0.509	0.106	0.714	0.291	0.673	0.272	0.768	0.293	0.703
	FedSA	0.275	0.766	0.098	0.771	0.314	0.645	0.311	0.749	0.340	0.662
	FedDistill	0.342	0.709	0.185	0.449	0.430	0.499	0.261	0.849	0.270	0.860

Table 2: PLIA attack performance on CIFAR-100 under varying α, M, and d. N/A: not applicable.

Attack	Algo.	Data Heterogeneity (α)				Number of Clients (M)				Prototype Dimension (d)			
		$\alpha = 0.1$		$\alpha = 1.0$		$M = 50$		$M = 100$		$d = 256$		$d = 1024$	
		TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑
Random	–	0.570	0.000	0.420	0.000	0.466	0.000	0.446	0.000	0.481	0.000	0.481	0.000
Uniform	–	0.539	–	0.354	–	0.411	–	0.385	–	0.431	–	0.431	–
Decaf	FedAvg	0.549	0.081	0.402	0.063	0.442	0.089	0.417	0.077	N/A	N/A	N/A	N/A
PGDA	FedAvg	**0.296**	**0.961**	0.305	**0.903**	0.392	0.709	0.382	0.330	N/A	N/A	N/A	N/A
P-InvDist	FedProto	0.445	0.850	0.279	0.817	0.322	0.815	0.305	0.748	0.342	0.826	0.335	0.869
	FedDistill	0.477	0.740	0.297	0.729	0.342	0.741	0.323	0.662	N/A	N/A	N/A	N/A
PLIA	FedProto	0.380	0.884	0.235	0.865	0.264	**0.893**	**0.247**	**0.875**	**0.286**	**0.860**	**0.290**	**0.916**
	FedDistill	0.326	0.873	**0.214**	0.801	**0.260**	0.792	0.263	0.747	N/A	N/A	N/A	N/A

Table 3: Comparison of DPP and DP-based defenses against PLIA.

Algo.	Defense	HAR		Amazon		CIFAR-10		CIFAR-100		Tiny IN	
		TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑
FedProto	No Defense	0.284	0.886	0.068	0.899	0.273	0.699	0.287	0.308	0.208	0.123
	IGPP	0.341	0.887	0.142	0.898	0.359	0.684	0.352	0.186	0.315	0.081
	VPDR	0.348	0.732	0.145	0.901	0.352	0.602	0.370	0.089	0.324	0.031
	DPP	**0.495**	0.878	**0.264**	0.899	**0.660**	0.700	**0.430**	0.307	**0.612**	0.121
FedDistill	No Defense	0.344	0.909	0.155	0.896	0.332	0.703	0.256	0.303	0.306	0.141
	IGPP	0.329	0.910	0.192	0.899	0.304	0.704	0.336	0.302	0.342	0.141
	VPDR	0.384	0.903	0.241	0.889	0.374	0.703	0.396	0.342	0.396	0.131
	DPP	**0.544**	0.911	**0.264**	0.895	**0.695**	0.703	**0.676**	0.304	**0.597**	0.143

further applies to heterogeneous architectures (Hete.), where it is generally less accurate than in the homogeneous setting but still outperforms Random and Uniform in all cases. Results for P-InvDist (Hete.), $\overline{\text{LDL}}^{\text{KL}}$, and standard deviations are provided in Appendix J.

Sensitivity Analysis. Table 2 evaluates PLIA on CIFAR-100 under varying heterogeneity, number of clients, and prototype dimension. PLIA outperforms the Random and Uniform baselines in all configurations and achieves the best results in most of them. The exceptions arise under varying heterogeneity: PGDA performs best on both metrics under the highly non-IID partitioning ($\alpha = 0.1$) and attains the highest LDL^{SC} at $\alpha = 1.0$, where PLIA still achieves the lowest LDL^{TV}. Table 8 (Appendix J) reports results for FedTGP and FedSA.

5.3 Defense Performance and Analysis

Overall Performance. Table 3 compares DPP with DP-based defenses, IGPP and VPDR, against PLIA. DPP provides the best protection in nearly all settings while keeping accuracy close to the undefended baseline. In contrast, the DP-based defenses offer only marginal protection and sometimes severely degrade accuracy on harder datasets such as Tiny ImageNet, suggesting that their noise is not designed to target the signal PLIA exploits. From the attacker's perspective, PLIA remains effective under DP: when IGPP and VPDR preserve accuracy comparable to no defense, $\overline{\text{LDL}}^{\text{TV}}$ changes only modestly and, in nearly all cases, remains below the Random and Uniform baselines in Table 1, indicating that label information still leaks. Results for FedTGP and FedSA, and for DPP on CIFAR-100 under varying conditions, are in Tables 10 and 11 (Appendix J). We further evaluate DPP against PLIA (multi-round), which averages the alignment signal across rounds (Appendix I).

Client-Level Analysis. Figure 4 visualizes the empirical and inferred label distributions of individual CIFAR-10 clients under FedProto, with classes sorted by decreasing training data proportion. Without DPP, PLIA's inferred distribution $\mathbf{q}_i$ (blue) closely tracks the ground-truth distribution $\hat{\mathbf{p}}_i$ (green). With DPP (orange), the inferred distribution deviates substantially from both the ground truth and the undefended estimate, showing that DPP effectively disrupts the geometric signal PLIA exploits. Results for additional clients and datasets are provided in Figures 8–12 in Appendix J.

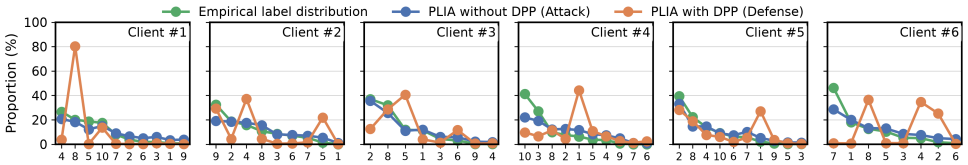
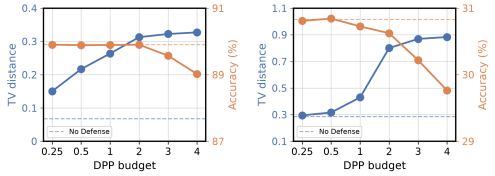
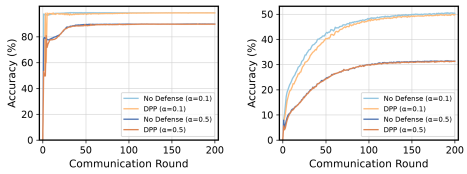


Figure 4: Per-client inferred label distributions with and without DPP on CIFAR-10 under FedProto.



(a) Amazon (b) CIFAR-100

Figure 5: Effect of perturbation budget on DPP.



(a) Amazon (b) CIFAR-100

Figure 6: Accuracy across comm. rounds.

Table 4: Ablation study on defense components against PLIA.

Defense	OD	CB	HAR		Amazon		CIFAR-10		CIFAR-100		Tiny IN	
			TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑
No Defense	-	-	0.284	0.886	0.068	0.899	0.273	0.699	0.287	0.308	0.208	0.123
RD-UB	✗	✗	0.312	0.886	0.103	0.899	0.322	0.700	0.299	0.309	0.237	0.124
RD-CB	✗	✓	0.317	0.888	0.140	0.899	0.353	0.700	0.302	0.309	0.246	0.123
OD-UB	✓	✗	0.316	0.883	0.147	0.899	0.330	0.698	0.324	0.306	0.266	0.122
OD-CB (DPP)	✓	✓	**0.495**	0.878	**0.264**	0.899	**0.660**	0.700	**0.430**	0.307	**0.612**	0.121

Budget Sensitivity Analysis. Figure 5 examines the privacy–utility trade-off of DPP under FedProto as the mean per-class budget varies, with dashed lines indicating the no-defense performance. $\overline{\text{LDL}}^{\text{TV}}$ increases with the budget and saturates at larger budgets, and a moderate budget (e.g., 2) already provides strong protection with negligible accuracy loss. Additional results are provided in Figure 13 (Appendix J).

Convergence and Efficiency. We examine whether DPP destabilizes training or adds significant overhead. As shown in Figure 6 for FedProto, DPP causes slight instability in early rounds, likely because prototypes are not yet well-formed, but converges to accuracy comparable to the undefended baseline across all settings (see Figure 14 in Appendix J for other datasets). DPP requires only one orthogonal projection per class, with negligible cost relative to local training, and leaves communication cost unchanged because it preserves prototype count and dimensionality.

DPP Component Ablation. Table 4 isolates two design components of DPP: direction (random, RD, vs. orthogonal, OD) and budget (uniform, UB, vs. class-wise, CB). All defense variants increase $\overline{\text{LDL}}^{\text{TV}}$ over the no-defense baseline, indicating reduced leakage. Either component alone yields only modest gains, whereas combining both in OD-CB (DPP) achieves by far the strongest protection.

6 Limitation and Conclusion

One limitation of PLIA is that, following the standard honest-but-curious assumption in the label distribution inference literature, it requires the server to observe each client's local prototypes individually. If secure aggregation is applied so that the server receives only aggregated prototypes, PLIA is no longer applicable in its current form. Within this setting, we identify a privacy vulnerability in prototype-based FL: the angular alignment between local and global prototypes can leak information about clients' label distributions. PLIA exploits this geometric signal to infer label distributions without access to model parameters or auxiliary data. To counter this, we propose DPP, which applies orthogonal, class-wise randomized perturbations to local prototypes with no communication overhead and minimal utility loss. These findings suggest that sharing prototypes alone does not guarantee the privacy of label distributions, and that effective defenses should account for the geometry of shared prototypes.

AI USE STATEMENT

In this work, we used generative AI tools to assist in writing and verifying the proof of Proposition 1, to provide feedback on experimental design, and to assist in implementing methods. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, assist with translation, or interpret results, and cleaning and reformatting datasets and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to draft and revise parts of the paper based on author-written English drafts, to edit the paper for readability, and to identify relevant literature. We have reviewed all AI-assisted work: the authors verified every step of the AI-assisted proof, AI-assisted code was verified and tested for correctness by two authors, and all AI-suggested references were checked against the original publications. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All settings needed to reproduce our experiments are given in Section 5.1 and Appendix F: datasets, model architectures, the Dirichlet partitioning with its rejection rule and fixed split seed, and the training and defense hyperparameters, including the DPP budget and the persistence of its per-class draws. PLIA and DPP are fully specified by Eqs. (8)–(11) and Algorithm 1 (Appendix E), the baseline attacks and defenses by Appendices G and H. All reported results are averaged over three random seeds, with standard deviations in Table 6; since GPU training kernels are not fully deterministic, individual runs may deviate slightly from the reported means within these standard deviations. The supplementary material provides anonymised source code for the defense module, the attack scorer and the partitioning script, together with the configuration files and commands used to produce every table and figure.

REFERENCES

Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *Advances in neural information processing systems*, 35:29677–29690, 2022.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 437–442, Bruges, Belgium, 2013.

Daeyoung Choi, Jihwan Shin, and Gyuejeong Lee. Prototype normalization: Optimizing prototype separation for heterogeneous federated learning. In *European Conference on Computer Vision*, pp. 116–134. Springer, 2026.

Yutong Dai, Zeyuan Chen, Junnan Li, Shelby Heinecke, Lichao Sun, and Ran Xu. Tackling data heterogeneity in federated learning with class prototypes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7314–7322, 2023.

Zhiyang Dai, Yansong Gao, Chunyi Zhou, Anmin Fu, Zhi Zhang, Minhui Xue, Yifeng Zheng, and Yuqing Zhang. Decaf: Data distribution decompose attack against federated learning. *IEEE Transactions on Information Forensics and Security*, 2024.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.

Boyu Fan, Siyang Jiang, Xiang Su, Sasu Tarkoma, and Pan Hui. A survey on model-heterogeneous federated learning: Problems, methods, and prospects. In *2024 IEEE international conference on big data (BigData)*, pp. 7725–7734. IEEE, 2024.

Yuhao Gu and Yuebin Bai. Ldia: Label distribution inference attack against federated learning in edge computing. *Journal of Information Security and Applications*, 74:103475, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021.

Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

Yangfan Jiang, Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Protecting label distribution in cross-silo federated learning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4828–4847. IEEE, 2024a.

Yangfan Jiang, Xinjian Luo, Yuncheng Wu, Xiaochen Zhu, Xiaokui Xiao, and Beng Chin Ooi. On data distribution leakage in cross-silo federated learning. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3312–3328, 2024b.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith 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.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations (ICLR)*, 2020.

Ashkan Khakzar, Soroosh Baselizadeh, Saurabh Khanduja, Christian Rupprecht, Seong Tae Kim, and Nassir Navab. Neural response interpretation through the lens of critical pathways. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13528–13538, 2021.

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

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Gyuejeong Lee and Daeyoung Choi. Class-wise federated averaging for efficient personalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1773–1782, 2025.

Gyuejeong Lee and Daeyoung Choi. Communication-efficient heterogeneous federated learning with sparse prototypes in resource-constrained environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 22787–22795, 2026.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Ziyao Liu, Jiale Guo, Wenzhuo Yang, Jiani Fan, Kwok-Yan Lam, and Jun Zhao. Privacy-preserving aggregation in federated learning: A survey. *IEEE Transactions on Big Data*, 2022.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys)*, pp. 165–172. ACM, 2013. doi: 10.1145/2507157.2507163.

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.

Sarthak Pati, Sourav Kumar, Amokh Varma, Brandon Edwards, Charles Lu, Liangqiong Qu, Justin J Wang, Anantharaman Lakshminarayanan, Shih-han Wang, Micah J Sheller, et al. Privacy preservation for federated learning in health care. *Patterns*, 5(7), 2024.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8432–8440, 2022a.

Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in neural information processing systems*, 35:19332–19344, 2022b.

Lei Wang, Jieming Bian, Letian Zhang, Chen Chen, and Jie Xu. Taming cross-domain representation variance in federated prototype learning with heterogeneous data domains. *Advances in Neural Information Processing Systems*, 37:88348–88372, 2024.

Li Wang, Qiang Wu, and Min Xu. Fedpcl-cdr: A federated prototype-based contrastive learning framework for privacy-preserving cross-domain recommendation. *Neural Networks*, pp. 108380, 2025.

Yuhua Wang, Qinnan Zhang, Xiaodong Li, Huan Zhang, Yifan Sun, Wangjie Qiu, Hainan Zhang, Yongxin Tong, and Zhiming Zheng. Taming noise-induced prototype degradation for privacy-preserving personalized federated fine-tuning. *arXiv preprint arXiv:2604.27833*, 2026.

Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

Jianqing Zhang, Yang Liu, Yang Hua, and Jian Cao. Fedtgp: Trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16768–16776, 2024a.

Jianqing Zhang, Xinghao Wu, Yanbing Zhou, Xiaoting Sun, Qiqi Cai, Yang Liu, Yang Hua, Zhenzhe Zheng, Jian Cao, and Qiang Yang. HtFLlib: A comprehensive heterogeneous federated learning library and benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2025. doi: 10.1145/3711896.3737379.

Jingyuan Zhang, Yiyang Duan, Shuaicheng Niu, Yang Cao, and Wei Yang Bryan Lim. Enhancing federated domain adaptation with multi-domain prototype-based federated fine-tuning. *arXiv preprint arXiv:2410.07738*, 2024b.

Chunyi Zhou, Yansong Gao, Anmin Fu, Kai Chen, Zhiyang Dai, Zhi Zhang, Minhui Xue, and Yuqing Zhang. Ppa: Preference profiling attack against federated learning. *arXiv preprint arXiv:2202.04856*, 2022.

Yanbing Zhou, Xiangmou Qu, Chenlong You, Jiyang Zhou, Jingyue Tang, Xin Zheng, Chunmao Cai, and Yingbo Wu. Fedsa: A unified representation learning via semantic anchors for prototype-based federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23009–23017, 2025.

Zhuangdi Zhu, Junyuan Hong, Steve Drew, and Jiayu Zhou. Resilient and communication efficient learning for heterogeneous federated systems. *Proceedings of machine learning research*, 162: 27504, 2022.

APPENDIX

A SAMPLE-LEVEL INTERPRETATION OF PROTOTYPE DISTILLATION

Eq. (5) expresses the prototype distillation regularizer in terms of class-level local and global prototypes, whereas in practice the regularization is implemented at the sample level. Specifically, for each local sample (x, y), its feature representation $\boldsymbol{c}_i(x)$ is encouraged to approach the corresponding global prototype $\bar{\boldsymbol{c}}_y^G$. For class j, the average sample-level regularization is

$$\mathcal{R}_{i,j}^F = \frac{1}{n_{i,j}} \sum_{(x,y)\in\mathcal{D}_{i,j}} \rho\big(\bar{\boldsymbol{c}}_j^G,\, \boldsymbol{c}_i(x)\big)\,, \tag{12}$$

where $\mathcal{D}_{i,j}$ denotes the local samples of class j and $n_{i,j} = |\mathcal{D}_{i,j}|$. Since ρ is the squared Euclidean distance and the local prototype $\bar{\boldsymbol{c}}_{i,j}^L$ is the mean of $\boldsymbol{c}_i(x)$ over $\mathcal{D}_{i,j}$, this term decomposes as

$$\mathcal{R}_{i,j}^F = \rho\big(\bar{\boldsymbol{c}}_j^G,\, \bar{\boldsymbol{c}}_{i,j}^L\big) + \frac{1}{n_{i,j}} \sum_{(x,y)\in\mathcal{D}_{i,j}} \rho\big(\bar{\boldsymbol{c}}_{i,j}^L,\, \boldsymbol{c}_i(x)\big)\,, \tag{13}$$

i.e., the class-level distance in Eq. (5) plus the within-class variance of the local features. Hence, $\mathcal{R}_{i,j}^F$ upper-bounds the class-level distance, and minimizing it drives the local prototype toward the global prototype. When the sample-level losses are aggregated over the entire local dataset $\mathcal{D}_i$ with $N_i = |\mathcal{D}_i|$, the overall regularizer becomes

$$\mathcal{R}_i^F = \frac{1}{N_i} \sum_{j\in\mathcal{Y}_i} \sum_{(x,y)\in\mathcal{D}_{i,j}} \rho\big(\bar{\boldsymbol{c}}_j^G,\, \boldsymbol{c}_i(x)\big) = \sum_{j\in\mathcal{Y}_i} \frac{n_{i,j}}{N_i}\, \mathcal{R}_{i,j}^F. \tag{14}$$

Compared with Eq. (5), which weights all classes equally, the implemented regularizer weights each class by its local sample proportion $n_{i,j}/N_i$. Consequently, within each client, classes with higher local prevalence receive stronger cumulative gradient pressure toward their global prototypes, which is consistent with the stronger local–global angular alignment observed in Figure 2. The same argument applies to the logit-level regularizer in Eq. (7), replacing feature representations with logits.

B SPIRAL DATASET GENERATION

The spiral dataset comprises six spiral-shaped clusters, one per class, designed to stress-test classifiers on nonlinear decision boundaries. For each class $k \in \{0, \cdots, 5\}$, we generate 5000 points as $\mathcal{C}_k = \{(x_{k,i}, y_{k,i}) \mid x_{k,i} = r_i \sin\omega_{k,i},\ y_{k,i} = r_i \cos\omega_{k,i},\ i \in [5000]\}$, where:

$$r_i = 1 + (i-1)\frac{9}{4999} \quad \text{and} \quad \omega_{k,i} = \frac{k}{3}\pi + (i-1)\frac{k}{3\times 4999}\pi + b_i \tag{15}$$

The radius r_i increases linearly from 1 to 10, while $\omega_{k,i}$ combines a class-dependent rotational offset $\frac{k}{3}\pi$ — separating the six spirals by $\frac{\pi}{3}$ radians — with a linearly increasing winding term. Gaussian noise $b_i \sim \mathcal{N}(0,1)$ introduces boundary overlap for more realistic decision boundaries. The dataset totals $30{,}000$ labeled samples in a two-dimensional feature space.

C PRIVACY METRIC FORMULATION

We define three metrics to quantify *Label Distribution Leakage* (LDL) between client i's empirical label distribution $\hat{\mathbf{p}}_i$ and the adversary's estimated distribution $\mathbf{q}_i$.

Total Variation Distance (TV) measures the maximum pointwise discrepancy between the two distributions:

$$\mathrm{LDL}_i^{\mathrm{TV}} = \frac{1}{2}\sum_{j=1}^{K} |\hat{p}_{i,j} - q_{i,j}|. \tag{16}$$

Kullback–Leibler Divergence (KL) penalizes discrepancies proportionally to the empirical distribution, emphasizing errors in high-probability classes:

$$\mathrm{LDL}_i^{\mathrm{KL}} = \sum_{j=1}^{K} \hat{p}_{i,j} \log \frac{\hat{p}_{i,j}}{q_{i,j}}. \tag{17}$$

Spearman Rank Correlation (SC) captures the monotonic agreement between the two distributions without assuming a specific divergence form:

$$\mathrm{LDL}_i^{\mathrm{SC}} = 1 - \frac{6\sum_{j=1}^{K} d_j^2}{K(K^2-1)}, \tag{18}$$

where $d_j = \mathrm{rank}(\hat{p}_{i,j}) - \mathrm{rank}(q_{i,j})$ is the rank difference for class j.

D Proof of Proposition 1

Proposition 1 (First-Order Optimality of Perpendicular Perturbation)**.** *Let $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ with $\boldsymbol{u} \neq \mathbf{0}$ and $\boldsymbol{v} \nparallel \boldsymbol{u}$, and let $\epsilon > 0$ satisfy $\epsilon \ll \|\boldsymbol{u}\|$. Among all perturbations $\boldsymbol{\delta}$ with $\|\boldsymbol{\delta}\| \leq \epsilon$, the perturbation that maximizes the angular distance between $\boldsymbol{u}' = \boldsymbol{u} + \boldsymbol{\delta}$ and $\boldsymbol{v}$ to first order in $\frac{\epsilon}{\|\boldsymbol{u}\|}$ is $\boldsymbol{\delta}^* = -\epsilon\frac{\boldsymbol{v}_\perp}{\|\boldsymbol{v}_\perp\|}$, where $\boldsymbol{v}_\perp = \boldsymbol{v} - \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|^2}\boldsymbol{u}$ is the component of $\boldsymbol{v}$ orthogonal to $\boldsymbol{u}$.*

Proof. Maximizing the angle between $\boldsymbol{u}' = \boldsymbol{u} + \boldsymbol{\delta}$ and $\boldsymbol{v}$ is equivalent to minimizing

$$\cos\theta(\boldsymbol{\delta}) = \frac{(\boldsymbol{u}+\boldsymbol{\delta})\cdot\boldsymbol{v}}{\|\boldsymbol{u}+\boldsymbol{\delta}\|\,\|\boldsymbol{v}\|},$$

since $\cos\theta$ is strictly decreasing on $[0,\pi]$. Assume $\|\boldsymbol{\delta}\| \leq \epsilon \ll \|\boldsymbol{u}\|$ and expand to first order in $\epsilon/\|\boldsymbol{u}\|$.

Using

$$\|\boldsymbol{u}+\boldsymbol{\delta}\| = \|\boldsymbol{u}\| + \frac{\boldsymbol{u}\cdot\boldsymbol{\delta}}{\|\boldsymbol{u}\|} + O\!\left(\frac{\epsilon^2}{\|\boldsymbol{u}\|}\right),$$

we obtain

$$\frac{1}{\|\boldsymbol{u}+\boldsymbol{\delta}\|} = \frac{1}{\|\boldsymbol{u}\|}\left(1 - \frac{\boldsymbol{u}\cdot\boldsymbol{\delta}}{\|\boldsymbol{u}\|^2}\right) + O\!\left(\frac{\epsilon^2}{\|\boldsymbol{u}\|^3}\right).$$

Substituting and expanding,

$$\cos\theta(\boldsymbol{\delta}) = \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|} + \frac{\boldsymbol{\delta}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|} - \frac{(\boldsymbol{u}\cdot\boldsymbol{v})(\boldsymbol{u}\cdot\boldsymbol{\delta})}{\|\boldsymbol{u}\|^3\|\boldsymbol{v}\|} + O\!\left(\frac{\epsilon^2}{\|\boldsymbol{u}\|^2}\right).$$

Decompose $\boldsymbol{v} = \boldsymbol{v}_\parallel + \boldsymbol{v}_\perp$, where $\boldsymbol{v}_\parallel = \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|^2}\boldsymbol{u}$. Since $\boldsymbol{v} \nparallel \boldsymbol{u}$, we have $\boldsymbol{v}_\perp \neq \mathbf{0}$. Then $\boldsymbol{\delta}\cdot\boldsymbol{v}_\parallel = \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|^2}(\boldsymbol{u}\cdot\boldsymbol{\delta})$, so the parallel terms cancel, yielding

$$\cos\theta(\boldsymbol{\delta}) = \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|} + \frac{\boldsymbol{\delta}\cdot\boldsymbol{v}_\perp}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|} + O\!\left(\frac{\epsilon^2}{\|\boldsymbol{u}\|^2}\right).$$

Thus, to first order, minimizing $\cos\theta(\boldsymbol{\delta})$ reduces to minimizing $\boldsymbol{\delta}\cdot\boldsymbol{v}_\perp$ subject to $\|\boldsymbol{\delta}\| \leq \epsilon$. By the Cauchy–Schwarz inequality,

$$\boldsymbol{\delta}\cdot\boldsymbol{v}_\perp \geq -\|\boldsymbol{\delta}\|\,\|\boldsymbol{v}_\perp\| \geq -\epsilon\|\boldsymbol{v}_\perp\|,$$

with equality when $\boldsymbol{\delta}$ is antiparallel to $\boldsymbol{v}_\perp$ and $\|\boldsymbol{\delta}\| = \epsilon$. Hence, to first order, the optimal perturbation is

$$\boldsymbol{\delta}^* = -\epsilon\frac{\boldsymbol{v}_\perp}{\|\boldsymbol{v}_\perp\|}.$$

By the same argument, $\boldsymbol{\delta}\cdot\boldsymbol{v}_\perp \leq \epsilon\|\boldsymbol{v}_\perp\|$, with equality when $\boldsymbol{\delta} = +\epsilon\frac{\boldsymbol{v}_\perp}{\|\boldsymbol{v}_\perp\|}$, which maximizes $\cos\theta(\boldsymbol{\delta})$ to first order. Therefore, $\pm\frac{\boldsymbol{v}_\perp}{\|\boldsymbol{v}_\perp\|}$ are the directions along which the cosine similarity changes most under the budget ϵ, with the first-order change bounded by $\epsilon\|\boldsymbol{v}_\perp\|/(\|\boldsymbol{u}\|\|\boldsymbol{v}\|)$ in magnitude. □

E Algorithm

Algorithm 1 summarizes how PLIA and DPP are integrated into FedProto.

Algorithm 1 FedProto with PLIA and DPP

Input: $\{\mathcal{D}_i\}_{i=1}^{M}$, $\{\boldsymbol{w}_i\}_{i=1}^{M}$, λ, ϵ_{total}; each client i keeps a store $\mathcal{S}_i$ of per-class draws (r_j, s_j), initially empty
Output: Trained local models $\boldsymbol{w}_i$ for $i = 1, \ldots, M$
Server executes:
Initialize global prototype set $\{\bar{\boldsymbol{c}}_j^G\}_{j=1}^{K}$.
for iteration $t = 1, \ldots, T$ **do**
  Sample a client subset $\mathcal{C}^t$
  for client $i \in \mathcal{C}^t$ in parallel **do**
    $\{\hat{\boldsymbol{c}}_{i,j}^L\}_{j\in\mathcal{Y}_i} \leftarrow$ **LocalUpdate**$\big(i, \{\bar{\boldsymbol{c}}_j^G\}_{j=1}^{K}\big)$
  for each client $i \in \mathcal{C}^t$ **do** ▷ **Attack: Inferring Label Distribution with PLIA**
    Estimate $\mathbf{q}_i$ from $\{\hat{\boldsymbol{c}}_{i,j}^L\}_{j\in\mathcal{Y}_i}$ and $\{\bar{\boldsymbol{c}}_j^G\}_{j\in\mathcal{Y}_i}$ using Eqs. (8) and (9)
  Compute $\bar{\boldsymbol{c}}_j^G$ for each class j by aggregating $\{\hat{\boldsymbol{c}}_{i,j}^L : i \in \mathcal{C}^t,\ j \in \mathcal{Y}_i\}$ using Eq. (3)

LocalUpdate$\big(i, \{\bar{\boldsymbol{c}}_j^G\}_{j=1}^{K}\big)$:
for each local epoch **do**
  for batch $b \in \mathcal{B}_i$ **do**
    Update model using the loss in Eqs. (4) and (5)
Compute $\{\bar{\boldsymbol{c}}_{i,j}^L\}_{j\in\mathcal{Y}_i}$ using Eq. (2)
for each class $j \in \mathcal{Y}_i$ with $(r_j, s_j) \notin \mathcal{S}_i$ **do** ▷ **First appearance of class j: draw once and reuse**
  Sample $r_j \sim \mathcal{U}(0,1)$ and $s_j \sim \mathcal{U}(\{-1,+1\})$ from client-private randomness and add (r_j, s_j) to $\mathcal{S}_i$
Set $\epsilon_j \leftarrow \epsilon_{\text{total}} \cdot r_j \big/ \sum_{k\in\mathcal{Y}_i} r_k$ for each $j \in \mathcal{Y}_i$ using the stored $(r_j, s_j) \in \mathcal{S}_i$
for each class $j \in \mathcal{Y}_i$ **do** ▷ **Defense: Manipulating Prototypes with DPP**
  Compute $\boldsymbol{v}_{\perp,j} = \bar{\boldsymbol{c}}_j^G - \frac{\bar{\boldsymbol{c}}_{i,j}^L \cdot \bar{\boldsymbol{c}}_j^G}{\|\bar{\boldsymbol{c}}_{i,j}^L\|^2} \bar{\boldsymbol{c}}_{i,j}^L$ using Eq. (11)
  Compute $\hat{\boldsymbol{c}}_{i,j}^L = \bar{\boldsymbol{c}}_{i,j}^L + s_j \epsilon_j \frac{\boldsymbol{v}_{\perp,j}}{\|\boldsymbol{v}_{\perp,j}\|}$ using Eq. (10)
return $\{\hat{\boldsymbol{c}}_{i,j}^L\}_{j\in\mathcal{Y}_i}$

F Experimental Details

This section describes the local training configuration and data partitioning used in our experiments. Unless otherwise specified, all results in the main paper follow the settings below. We provide our code in the supplementary materials to support reproducibility.

Local Training Configuration. All experiments use 20 clients with full participation, 200 communication rounds, one local epoch per round, a learning rate of 0.01, batch size 32, and a prototype dimension of $d = 512$. The prototype regularization weight is $\lambda = 1$ for FedProto, FedSA and FedDistill and $\lambda = 10$ for FedTGP. Clients compute their local prototypes on the training data after local training and upload them once per round, and the attack is evaluated in every round. Accuracy is the best test accuracy over rounds, pooled over the test sets of all clients. Every configuration is run with three seeds; we report the mean, and Table 6 gives the standard deviations of the attack metrics. Each run uses a single NVIDIA RTX PRO 6000 Blackwell GPU.

Data Partition. Data heterogeneity is induced by a per-class Dirichlet partition with concentration $\alpha = 0.5$ over 20 clients; lower α gives more skewed class distributions, and we report ablations for $\alpha \in \{0.1, 0.5, 1.0\}$. The samples assigned to each client are split $75\%/25\%$ into train and test sets. The partition is generated once with a fixed seed and reused in all runs, so every method, attack and defense sees identical client data. A partition in which some client receives fewer than $\min\left(\frac{B}{1-r}, \frac{N}{2M}\right)$ samples, with $B = 10$, $r = 0.75$, N the dataset size and M the number of clients, is rejected and the Dirichlet draw is repeated.

G Prototype-Based Baseline Attack Method: P-InvDist

As discussed in Section 4.2, the prototype regularization in Eq. (5) uses Euclidean distance, making it a natural alternative to cosine similarity for scoring class prevalence. We therefore construct

P-InvDist, a variant of PLIA that replaces the alignment score with the inverse Euclidean distance between local and global prototypes, while keeping all other components identical.

Formulation. Following Section 3.3, classes with more local samples receive stronger cumulative gradient pressure, which pulls their local prototypes closer to the global prototypes in Euclidean space. P-InvDist directly measures this proximity. For client i and class $j \in \mathcal{Y}_i$ at round t, P-InvDist replaces the alignment score in Eq. (8) with

$$\Delta_{i,j}^{t} = \frac{1}{\left\| \bar{\boldsymbol{c}}_j^{G,t-1} - \bar{\boldsymbol{c}}_{i,j}^{L,t} \right\|_2 + \eta}, \tag{19}$$

where $\eta > 0$ is a small constant that prevents division by zero when a local prototype coincides with the global one. The inferred label distribution $\mathbf{q}_i^t$ is then obtained by normalizing these scores over $\mathcal{Y}_i$ as in Eq. (9). For logit-based algorithms such as FedDistill, the prototypes in Eq. (19) are the class-wise logits regularized in Eq. (7). Like PLIA, P-InvDist uses only the prototypes the server observes in a single round and requires no auxiliary data or model access.

Comparison with PLIA. The two scoring rules differ in their sensitivity to prototype magnitude. Euclidean distance reflects both the angle and the norms of the two prototypes, so differences in prototype norm across classes or clients, which are common under heterogeneous model architectures, directly affect the score even when they are unrelated to the local sample count. In contrast, cosine similarity depends only on the angle, capturing the directional pull toward the global prototype while remaining insensitive to magnitude. We conjecture that this difference contributes to P-InvDist's weaker and less consistent performance in Tables 1 and 5.

H Baseline Defense Methods

For each class j, let $\boldsymbol{u} = \bar{\boldsymbol{c}}_{i,j}^{L,t}$ and $\boldsymbol{v} = \bar{\boldsymbol{c}}_j^{G,t-1}$ denote the local and global prototype, respectively. The perturbed prototype is given by:

$$\hat{\boldsymbol{u}} = \boldsymbol{u} + \epsilon_j \cdot \boldsymbol{\delta}_j, \tag{20}$$

where $\boldsymbol{\delta}_j \in \mathbb{R}^d$ with $\|\boldsymbol{\delta}_j\| = 1$ is the perturbation direction and $\epsilon_j > 0$ is the per-class budget. The Orthogonal Direction and Class-Wise Budget components of DPP are described in Section 4.3. The remaining ablation baselines differ in their choices of $\boldsymbol{\delta}_j$ and ϵ_j as follows; all of them spend the same total budget ϵ_{total} per client as DPP.

Random Direction (RD). The perturbation direction is sampled uniformly from the unit sphere:

$$\boldsymbol{\delta}_j = \frac{\boldsymbol{\xi}_j}{\|\boldsymbol{\xi}_j\|}, \quad \boldsymbol{\xi}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d). \tag{21}$$

This direction-agnostic baseline does not exploit the geometric structure of the local–global prototype alignment. RD-CB combines it with the class-wise budgets of DPP, using the same persistent draws r_j (Section 4.3), so that RD-CB and DPP differ only in the direction.

Uniform Budget Allocation (UB). The total budget is split equally across the classes held by the client:

$$\epsilon_j = \frac{\epsilon_{\text{total}}}{|\mathcal{Y}_i|}, \quad \forall j \in \mathcal{Y}_i. \tag{22}$$

Equal scaling shifts every prototype by the same magnitude, preserving the relative angular alignment between classes and leaving the inferred $\mathbf{q}_i$ correlated with the true empirical distribution $\hat{\mathbf{p}}_i$. OD-UB combines this allocation with the orthogonal direction of Eq. (11) applied with a fixed sign, $\boldsymbol{\delta}_j = -\boldsymbol{v}_\perp / \|\boldsymbol{v}_\perp\|$, so that every class is rotated away from its global prototype.

For IGPP and VPDR, we follow the notation of the original paper (Wang et al., 2026); in the following two paragraphs, symbols such as ϵ, δ, and Δ_j are local to these descriptions and differ from their meanings elsewhere in this paper.

Isotropic Gaussian Prototype Perturbation (IGPP). IGPP (Wang et al., 2026) clips each local feature to the L_2 radius R before averaging and releases the clipped class mean with isotropic

Gaussian noise calibrated to (ϵ, δ)-local differential privacy over the T rounds:

$$\hat{\boldsymbol{u}} = \boldsymbol{u}_R + \boldsymbol{n}_j, \quad \boldsymbol{n}_j \sim \mathcal{N}\big(\mathbf{0}, (\sigma\Delta_j)^2\mathbf{I}_d\big), \quad \Delta_j = \frac{2R}{n_{i,j}}, \quad \sigma = \frac{2\sqrt{T\ln(1/\delta)}}{\epsilon}, \tag{23}$$

where $\boldsymbol{u}_R$ is the clipped mean. The noise grows inversely with the class count, so rare classes are perturbed most; it hides individual samples rather than the label distribution and does not exploit the local–global prototype geometry. We use $\delta = 10^{-5}$, $T = 200$, and $(R, \epsilon) = (0.5, 512)$ for FedProto, $(5, 512)$ for FedSA, and the reference $(5, 128)$ for FedTGP and FedDistill, chosen per algorithm by a grid search over $R \in \{0.5, 1, 2, 5\}$ and $\epsilon \in \{32, 128, 512\}$ on shortened runs. We selected these settings to preserve model utility.

Variance-aware Prototype Perturbation with Distillation-based Representation Recovery (VPDR). VPDR (Wang et al., 2026) extends IGPP by privately selecting, with a fraction r of the budget, a discriminative subset A of $\lceil \rho d \rceil$ coordinates and its complement B, which are clipped to $R_A = R\sqrt{d_A/d}$ and $R_B = R\sqrt{d_B/d}$ and perturbed separately:

$$\hat{\boldsymbol{u}}_A = \boldsymbol{u}_{R,A} + \boldsymbol{n}_{j,A}, \quad \hat{\boldsymbol{u}}_B = \boldsymbol{u}_{R,B} + \boldsymbol{n}_{j,B}, \quad \boldsymbol{n}_{j,G} \sim \mathcal{N}\big(\mathbf{0}, (\sigma_G\Delta_j\kappa_G)^2\mathbf{I}_{d_G}\big),\ G \in \{A, B\}, \tag{24}$$

where $\kappa_G = \sqrt{d_G/d}$ and $\sigma_A^{-2} + \sigma_B^{-2} = \sigma_{\text{ref}}^{-2}$ with σ_{ref} calibrated as in Eq. (23) to the remaining budget $(1-r)\epsilon$, so that the discriminative coordinates receive less noise under the same guarantee. Local training further adds a distillation term $\lambda_1\mathcal{L}_{\text{KD}}$ from an exponential-moving-average teacher to recover from the noisy prototypes. We use the authors' defaults ($r = 0.1$, $\rho = 0.2$, $\gamma = 0.05$, $\beta = 0.999$, $\tau = 4$, $\lambda_1 = 0.05$) and the same (R, ϵ, δ, T) as IGPP. Like IGPP, it allocates noise based on coordinate discriminability, not on the angular signal PLIA exploits.

I Multi-Round Attack against DPP

Since DPP perturbs prototypes at every round, an attacker may attempt to weaken its effect by aggregating observations across rounds. We therefore evaluate DPP against a multi-round option of PLIA, denoted PLIA (multi-round), which averages the alignment signal over rounds before inference.

Attack formulation. The default PLIA computes the cosine similarity between local and global prototypes, infers the label distribution independently at each round, and reports $\overline{\text{LDL}}^{\text{TV}}$ averaged over all rounds. In contrast, at each round, PLIA (multi-round) averages the cosine similarity for each client and class from the first round up to the current round and infers the label distribution from this average using the same scoring and normalization. Its reported $\overline{\text{LDL}}^{\text{TV}}$ is the value at the final round, where the similarity is averaged over all rounds. Averaging is a natural strategy against a randomized defense, as the class-prevalence signal persists throughout training. At the same time, the effect of a perturbation may vary across rounds, e.g., because DPP recomputes the perturbation direction from evolving prototypes. Since the server observes prototypes at every round under our threat model, PLIA (multi-round) tests whether aggregating observations over time can weaken DPP.

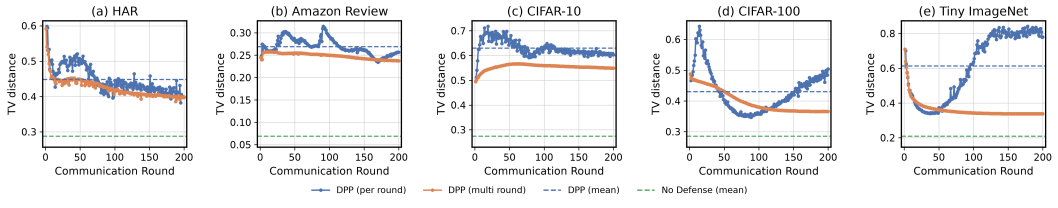


Figure 7: DPP result across rounds under FedProto. DPP (per round): default PLIA at each round. DPP (multi-round): PLIA (multi-round) with similarity averaged up to each round. DPP (mean) and No Defense (mean): round-averaged $\overline{\text{LDL}}^{\text{TV}}$ under the default PLIA with and without DPP.

Results. Figure 7 shows $\overline{\text{LDL}}^{\text{TV}}$ of DPP under both attacks across communication rounds under FedProto. Under the default PLIA, the per-round $\overline{\text{LDL}}^{\text{TV}}$ fluctuates considerably. In contrast, under PLIA (multi-round), it changes smoothly and gradually converges, since the similarity is averaged from the first round up to the current round. PLIA (multi-round) reduces DPP protection on all datasets, most notably on Tiny ImageNet. Nevertheless, DPP maintains a higher $\overline{\text{LDL}}^{\text{TV}}$ than the undefended baseline on all datasets, indicating that it still reduces leakage, although less than under the default PLIA.

J Additional Results

This section provides additional experimental results complementing the main paper. Table 5 reports P-InvDist attack performance under heterogeneous model architectures, and Table 6 reports the standard deviations of the attack metrics over three random seeds, confirming the stability of our results. Table 7 reports $\overline{\text{LDL}}^{\text{KL}}$ under both homogeneous and heterogeneous architectures, complementing Table 1, and Table 9 reports $\overline{\text{LDL}}^{\text{KL}}$ for the sensitivity analysis in Table 2, which Table 8 further extends to FedTGP and FedSA. Table 10 extends the defense comparison in Table 3 to FedTGP and FedSA, and Table 11 evaluates DPP on CIFAR-100 under varying α, M, and d. Figures 8–12 visualize per-client inferred label distributions with and without DPP on all five datasets under FedProto. Finally, Figures 13 and 14 show the effect of the perturbation budget on DPP and the accuracy curves across communication rounds, respectively, on the remaining datasets, complementing Figures 5 and 6.

Table 5: P-InvDist attack performance under heterogeneous model architectures (TV: $\overline{\text{LDL}}^{\text{TV}}$, SC: $\overline{\text{LDL}}^{\text{SC}}$). Results for PLIA (Hete.) and the Random and Uniform baselines are reported in Table 1.

Attack	Algo.	HAR		Amazon		CIFAR-10		CIFAR-100		Tiny IN	
		TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑
P-InvDist (Hete.)	FedProto	0.461	-0.338	0.220	0.123	0.321	0.790	0.360	0.627	0.364	0.631
	FedTGP	0.337	0.575	0.163	0.600	0.334	0.644	0.336	0.742	0.357	0.679
	FedSA	0.450	-0.062	0.160	0.569	0.353	0.576	0.358	0.675	0.362	0.620
	FedDistill	0.412	0.332	0.183	0.243	0.373	0.472	0.331	0.744	0.354	0.723

Table 6: Standard deviations of attack performance (TV: $\overline{\text{LDL}}^{\text{TV}}$, SC: $\overline{\text{LDL}}^{\text{SC}}$) over three random seeds, corresponding to Tables 1 and 5.

Attack	Algo.	HAR		Amazon		CIFAR-10		CIFAR-100		Tiny IN	
		TV	SC	TV	SC	TV	SC	TV	SC	TV	SC
Decaf	FedAvg	0.002	0.010	0.003	0.014	0.004	0.019	0.001	0.003	0.000	0.001
PGDA	FedAvg	0.017	0.017	0.005	0.041	0.033	0.027	0.007	0.003	0.007	0.007
P-InvDist (Homo.)	FedProto	0.002	0.033	0.002	0.023	0.001	0.009	0.001	0.000	0.000	0.000
	FedTGP	0.009	0.016	0.001	0.004	0.002	0.005	0.000	0.003	0.001	0.005
	FedSA	0.004	0.014	0.002	0.112	0.004	0.004	0.001	0.003	0.000	0.004
	FedDistill	0.001	0.017	0.001	0.005	0.000	0.005	0.000	0.003	0.006	0.053
PLIA (Homo.)	FedProto	0.003	0.011	0.001	0.044	0.003	0.008	0.002	0.002	0.001	0.001
	FedTGP	0.020	0.037	0.004	0.014	0.002	0.009	0.001	0.002	0.002	0.006
	FedSA	0.010	0.006	0.004	0.003	0.016	0.027	0.003	0.005	0.001	0.001
	FedDistill	0.003	0.005	0.010	0.055	0.005	0.008	0.001	0.002	0.013	0.046
P-InvDist (Hete.)	FedProto	0.002	0.009	0.001	0.020	0.001	0.011	0.000	0.001	0.000	0.001
	FedTGP	0.003	0.009	0.000	0.003	0.003	0.008	0.000	0.003	0.001	0.006
	FedSA	0.002	0.012	0.006	0.091	0.003	0.025	0.001	0.007	0.000	0.003
	FedDistill	0.001	0.010	0.000	0.011	0.002	0.007	0.000	0.000	0.004	0.014
PLIA (Hete.)	FedProto	0.001	0.021	0.001	0.074	0.002	0.009	0.001	0.007	0.000	0.005
	FedTGP	0.008	0.023	0.009	0.015	0.005	0.015	0.001	0.001	0.003	0.004
	FedSA	0.008	0.023	0.005	0.033	0.014	0.013	0.002	0.003	0.001	0.002
	FedDistill	0.006	0.004	0.039	0.072	0.006	0.012	0.002	0.002	0.009	0.018

Table 7: PLIA attack performance on various datasets (KL: $\overline{\text{LDL}}^{\text{KL}}$). **Bold**: best per column.

Attack	Algo.	HAR	Amazon	CIFAR-10	CIFAR-100	Tiny IN
		KL↓	KL↓	KL↓	KL↓	KL↓
Random	–	0.798	0.371	0.860	0.872	0.880
Uniform	–	0.543	0.216	0.574	0.567	0.574
Decaf	FedAvg	0.681	0.200	0.572	0.741	0.769
PGDA	FedAvg	0.378	0.172	0.365	0.346	**0.112**
P-InvDist (Homo.)	FedProto	0.499	0.130	0.398	0.357	0.301
	FedTGP	0.257	0.103	0.205	0.428	0.328
	FedSA	0.521	0.101	0.489	0.448	0.477
	FedDistill	0.514	0.155	0.583	0.416	0.456
PLIA (Homo.)	FedProto	0.270	**0.033**	0.236	0.263	0.152
	FedTGP	0.317	0.088	**0.117**	0.339	0.192
	FedSA	**0.190**	0.042	0.218	0.264	0.427
	FedDistill	0.508	0.364	0.363	**0.224**	0.346
P-InvDist (Hete.)	FedProto	0.625	0.224	0.328	0.404	0.411
	FedTGP	0.353	0.131	0.338	0.364	0.406
	FedSA	0.594	0.142	0.387	0.420	0.425
	FedDistill	0.525	0.192	0.440	0.356	0.398
PLIA (Hete.)	FedProto	0.481	0.140	0.210	0.346	0.364
	FedTGP	0.325	0.068	0.288	0.256	0.303
	FedSA	0.239	0.072	0.342	0.329	0.388
	FedDistill	0.566	0.568	0.698	0.236	0.256

Table 8: PLIA attack performance on CIFAR-100 under varying α, M, and d for FedTGP and FedSA (TV: $\overline{\text{LDL}}^{\text{TV}}$, SC: $\overline{\text{LDL}}^{\text{SC}}$). Results for the other algorithms and baselines are reported in Table 2.

Attack	Algo.	Data Heterogeneity (α)				Number of Clients (M)				Prototype Dimension (d)			
		$\alpha = 0.1$		$\alpha = 1.0$		$M = 50$		$M = 100$		$d = 256$		$d = 1024$	
		TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑	TV↓	SC↑
P-InvDist	FedTGP	0.437	0.769	0.309	0.589	0.365	0.639	0.349	0.573	0.373	0.658	0.380	0.696
	FedSA	0.494	0.663	0.307	0.657	0.345	0.765	0.317	0.745	0.370	0.704	0.376	0.713
PLIA	FedTGP	0.337	0.751	0.286	0.602	0.342	0.572	0.346	0.425	0.328	0.669	0.328	0.701
	FedSA	0.357	0.897	0.238	0.843	0.269	0.898	0.254	0.885	0.291	0.850	0.288	0.911

Table 9: PLIA attack performance on CIFAR-100 under varying α, M, and d (KL: $\overline{\text{LDL}}^{\text{KL}}$). **Bold**: best per column. N/A: not applicable.

Attack	Algo.	Data Heterogeneity (α)		Number of Clients (M)		Prototype Dimension (d)	
		$\alpha = 0.1$	$\alpha = 1.0$	$M = 50$	$M = 100$	$d = 256$	$d = 1024$
		KL↓	KL↓	KL↓	KL↓	KL↓	KL↓
Random	–	1.140	0.692	0.815	0.752	0.872	0.872
Uniform	–	0.837	0.387	0.510	0.448	0.567	0.567
Decaf	FedAvg	0.974	0.590	0.685	0.623	N/A	N/A
PGDA	FedAvg	**0.282**	0.285	0.457	0.437	N/A	N/A
P-InvDist	FedProto	0.578	0.244	0.316	0.285	0.367	0.352
	FedTGP	0.564	0.300	0.403	0.366	0.429	0.443
	FedSA	0.737	0.303	0.370	0.315	0.441	0.455
	FedDistill	0.679	0.280	0.361	0.323	N/A	N/A
PLIA	FedProto	0.440	0.178	**0.220**	**0.194**	**0.264**	**0.267**
	FedTGP	0.379	0.272	0.378	0.388	0.353	0.348
	FedSA	0.420	0.188	0.233	0.207	0.279	0.270
	FedDistill	0.356	**0.161**	0.229	0.227	N/A	N/A

Table 10: Defense performance comparison under FedTGP and FedSA (TV: $\overline{\text{LDL}}^{\text{TV}}$, higher indicates stronger protection; Acc.: accuracy). **Bold**: best protection per algorithm and dataset.

Algo.	Defense	HAR		Amazon		CIFAR-10		CIFAR-100		Tiny IN	
		TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑	TV↑	Acc.↑
FedTGP	No Defense	0.280	0.905	0.111	0.901	0.179	0.730	0.322	0.347	0.240	0.112
	IGPP	0.343	0.851	0.086	0.898	0.304	0.714	0.301	0.282	0.269	0.142
	VPDR	0.348	0.775	0.166	0.900	0.307	0.703	0.343	0.195	0.334	0.021
	DPP	**0.419**	0.907	**0.232**	0.900	**0.635**	0.732	**0.697**	0.350	**0.714**	0.118
FedSA	No Defense	0.238	0.922	0.069	0.892	0.258	0.715	0.283	0.326	0.357	0.144
	IGPP	0.344	0.922	0.144	0.893	0.376	0.718	0.318	0.304	0.319	0.109
	VPDR	**0.350**	0.912	**0.151**	0.900	0.348	0.722	**0.347**	0.245	0.322	0.076
	DPP	0.270	0.922	0.104	0.892	**0.447**	0.716	0.327	0.326	**0.810**	0.143

Table 11: DPP defense performance on CIFAR-100 under FedProto with varying α, M, and d (TV: $\overline{\text{LDL}}^{\text{TV}}$, SC: $\overline{\text{LDL}}^{\text{SC}}$; Acc.: accuracy).

Defense	Data Heterogeneity (α)						Number of Clients (M)						Prototype Dimension (d)					
	$\alpha = 0.1$			$\alpha = 1.0$			$M = 50$			$M = 100$			$d = 256$			$d = 1024$		
	TV↑	SC↓	Acc.↑	TV↑	SC↓	Acc.↑	TV↑	SC↓	Acc.↑	TV↑	SC↓	Acc.↑	TV↑	SC↓	Acc.↑	TV↑	SC↓	Acc.↑
No Defense	0.380	0.884	0.500	0.235	0.865	0.253	0.264	0.893	0.115	0.247	0.875	0.117	0.286	0.860	0.307	0.290	0.916	0.309
DPP	**0.764**	**0.419**	0.493	**0.334**	**0.581**	0.252	**0.349**	**0.666**	0.111	**0.339**	**0.622**	0.098	**0.575**	**0.532**	0.305	**0.359**	**0.683**	0.307

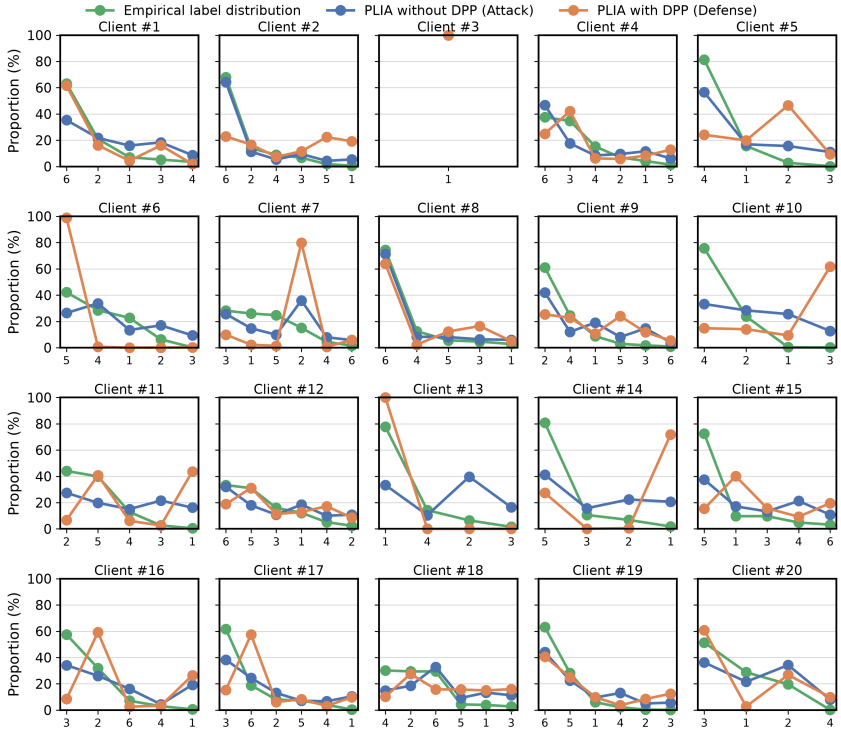


Figure 8: Per-client inferred label distributions with and without DPP on HAR under FedProto.

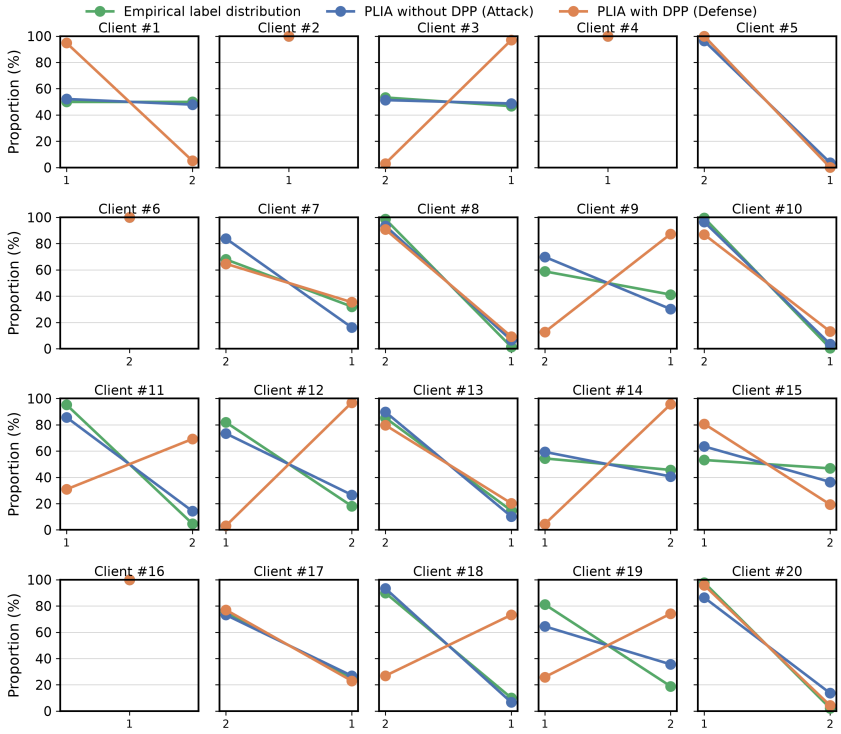


Figure 9: Per-client inferred label distributions with and without DPP on Amazon Review under FedProto.

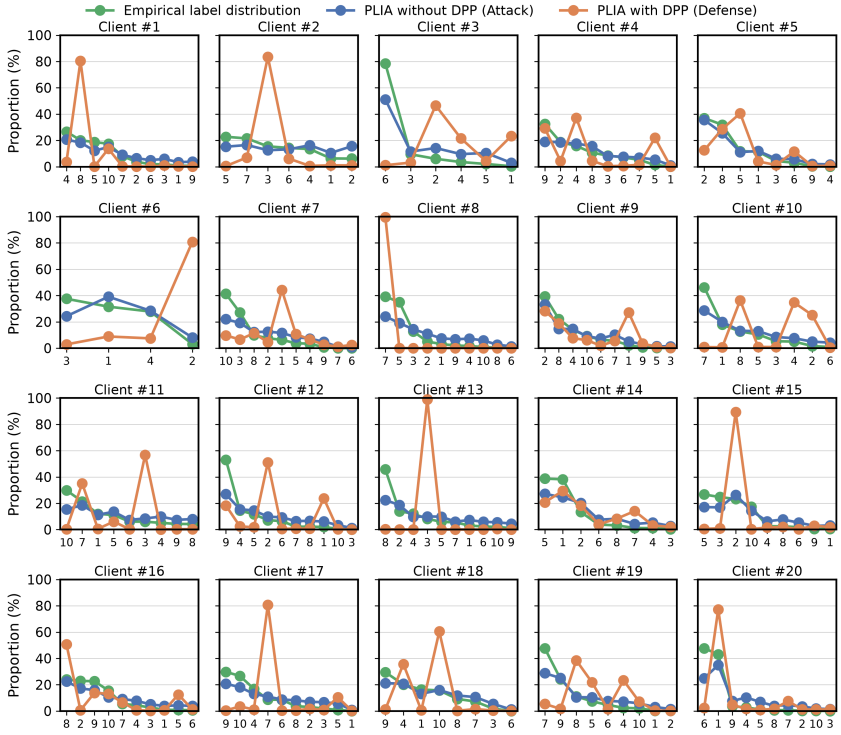


Figure 10: Per-client inferred label distributions with and without DPP on CIFAR-10 under FedProto.

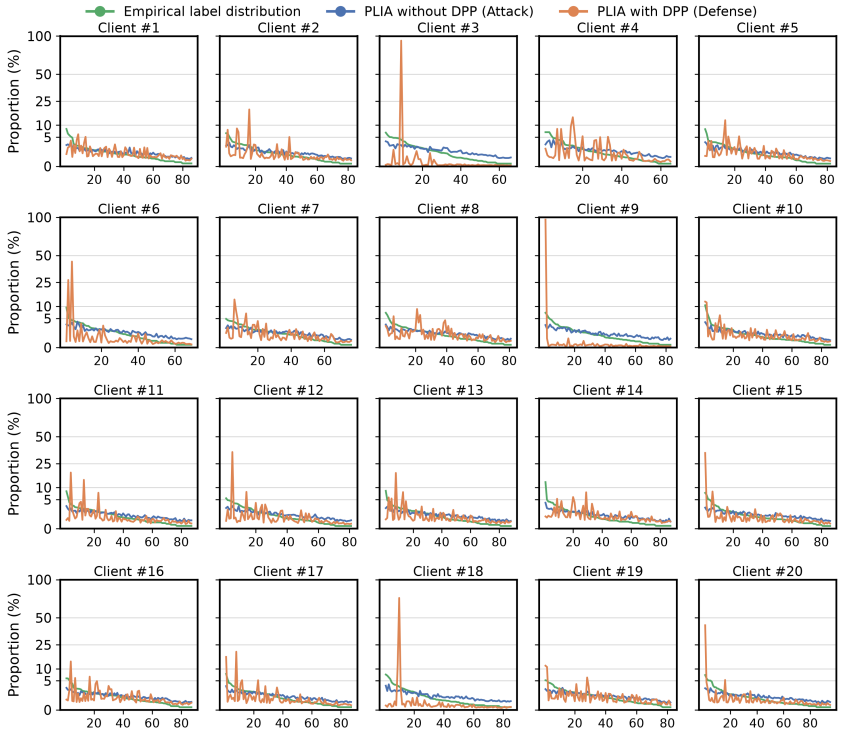


Figure 11: Per-client inferred label distributions with and without DPP on CIFAR-100 under FedProto.

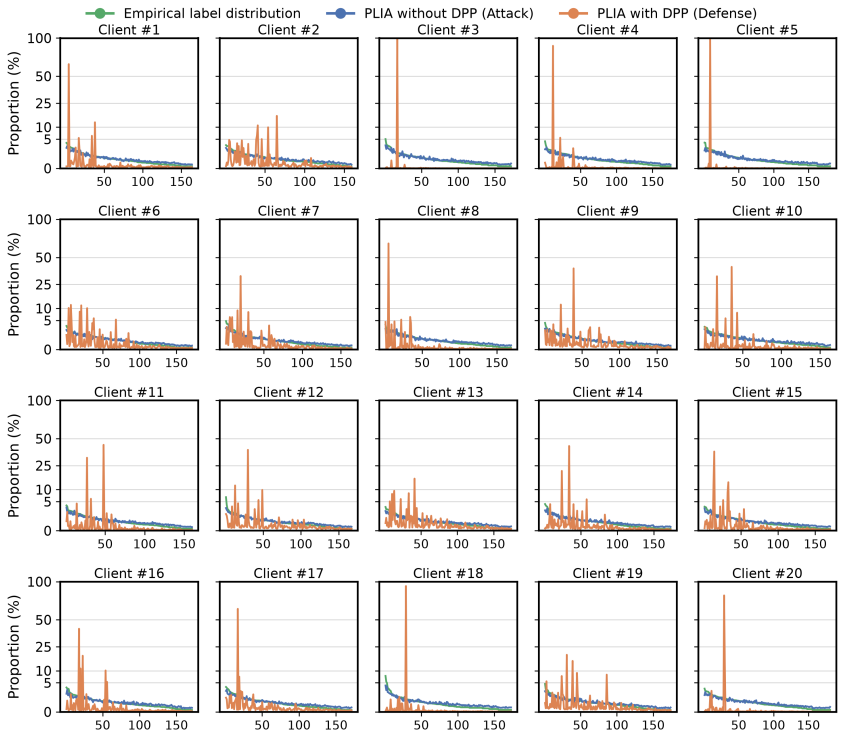
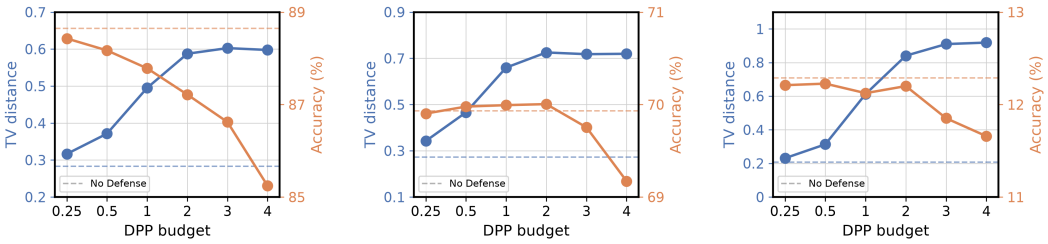
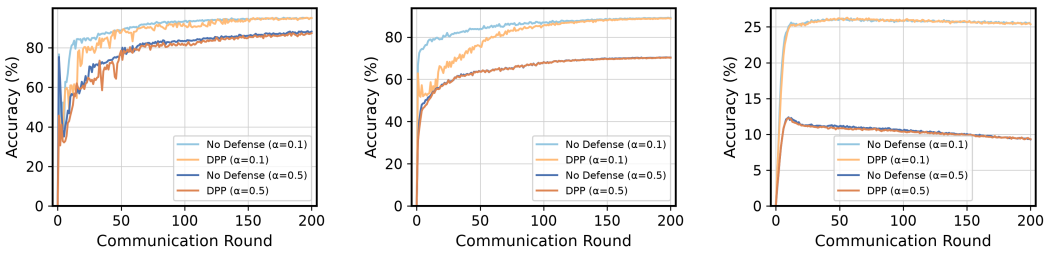


Figure 12: Per-client inferred label distributions with and without DPP on Tiny ImageNet under FedProto.



(a) HAR (b) CIFAR-10 (c) Tiny ImageNet

Figure 13: Effect of perturbation budget on DPP under FedProto.



(a) HAR (b) CIFAR-10 (c) Tiny ImageNet

Figure 14: Accuracy across communication rounds under FedProto.