Synthetic Anchor-Assisted Prototype Alignment for Heterogeneous Federated Learning

Zhihao Hao[1,2*] **Bob Zhang**[3]
[1]School of Computer Science and Artificial Intelligence &
Beijing Key Laboratory of Commercial Data Security Protection
and Intelligent Governance,
Beijing Technology and Business University, Beijing, China
[2]Beijing Key Laboratory of Applied Statistics and Digital Regulation &
Academy for Interdisciplinary Studies,
Beijing Technology and Business University, Beijing, China
[3]Dept. of Artificial Intelligence, University of Macau, Macau
zhihao.hao@btbu.edu.cn, bobzhang@um.edu.mo

Abstract

Heterogeneous federated learning must connect clients that run different architectures and hold skewed labels. Their parameters cannot be averaged, and their local class prototypes drift apart. We propose SATPL, which gives all clients a shared semantic coordinate system built from two references. The first is a set of fixed synthetic anchors. A language model describes each class, a diffusion model renders the descriptions, and a frozen CLIP encoder turns the images into anchors. The second is a global prototype bank that the server rebuilds every round. Clients align their features with the anchors and classify them with the bank, so knowledge from other clients enters every local loss. The server weights each uploaded prototype by its agreement with the class anchor. We prove that this rule suppresses unreliable uploads under a positive similarity gap and bound the resulting shift in the broadcast direction. In fixed-seed results on five benchmarks, SATPL ranks first and also leads four recent baselines. Because their run-level records are not available, we add a fully logged Caltech-10 study. There, anchor alignment and anchor weighting each raise pooled local accuracy, and SATPL gains 7.5 points over head averaging with about 1% of its message size, while FedProto and local training stay slightly ahead. An optional ledger keeps an audit trail of aggregation metadata, independent of training and weighting.

1 Introduction

Federated learning lets clients train together while their data stay local [1]. Parameter averaging works well when all clients share one model structure. In practice, devices with different resource limits often run different backbones. Their parameter tensors need not share shapes or meanings, so direct averaging cannot connect them. Heterogeneous federated learning therefore needs an interface that carries task knowledge across architectures [2].

Class prototypes provide such an interface. FedProto exchanges class summaries, and FedGen transfers knowledge through a learned generator [3, 4]. In both cases, the shared information comes from the federation itself. Under label skew, a class prototype then depends on which clients take part and how many examples they hold. A common vector size makes uploads compatible, but it does not make their directions comparable. FedLSA learns semantic anchors with hyperspherical

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

contrast, and FedSA updates semantic anchors from client prototypes [5, 6]. Both show that the way a shared reference is formed and updated matters.

SATPL builds a class reference before federated training starts. A language model writes a visual description for each class name. A diffusion model renders these descriptions, and a frozen encoder maps the images to class anchors. Every client receives the same anchors. The anchors give each client a fixed target for local alignment and give the server a reference for weighting uploads. They serve as class-level semantic references shared by all clients.

A second reference carries knowledge between clients. After each round, the server rebuilds a global prototype bank from the uploaded class prototypes. Clients then use this bank as the classifier in their next local loss. The bank stays fixed during a local update, but its vectors come from other clients' data. In this way, other clients' prototypes shape every local objective. The fixed anchors guide this exchange, while the changing bank tracks the federation's current representations. Figure 1 shows how the two references work together.

Recent methods transfer knowledge through private and proxy models, trainable global prototypes, or sparse prototype messages [7–9]. Generative models and blockchains also support heterogeneous collaboration. FRIL fuses heterogeneous models through a blockchain for robust food recognition [10]. FusionIL uses large language models to synthesize training data, validates the data on a consortium blockchain, and lets clients with structurally different models learn from it together with their private data [11]. SATPL uses external models differently. They run once, before training, and turn class descriptions into fixed anchors, so no synthetic sample enters local training. This prior is a declared part of the method, and we test it with controlled anchor variants.

Our work makes three contributions. We specify a complete training interface in which fixed synthetic anchors guide alignment and server weighting, and an evolving global bank serves as every client's classifier. We analyze anchor-consistency weighting and bound both the weight of unreliable uploads and the change in the broadcast direction. We also evaluate SATPL on five benchmarks, with anchor-construction controls and recent baselines, and we add a fully logged Caltech-10 study. Throughout, clients exchange only class-indexed prototypes, about 1.2% of the bytes of head averaging in that study. An optional permissioned ledger can keep an audit trail of anchor commitments and aggregation metadata, independent of training and weighting.

The rest of this paper is organized as follows. Section 2 presents the SATPL framework, from anchor construction to prediction. Section 3 analyzes anchor-consistency weighting. Section 4 reports the benchmark comparisons, the anchor controls and ablations, and the logged Caltech-10 study. Section 5 concludes the paper.

2 Framework

2.1 Problem and shared representation

We consider a closed-set task with K clients and C shared class labels. Client k holds private data $\mathcal{D}_k$ and observes classes $\mathcal{C}_k \subseteq \{1, \dots, C\}$. Its backbone f_k may differ from other clients' backbones in architecture and feature dimension. A local projection head g_k maps these features to $\mathbb{R}^d$. Neither backbone nor projection parameters are averaged. Clients exchange class-indexed vectors in the common space.

The common label mapping is part of the task definition. Each bank column and anchor must refer to the same class for every client. A client need not hold examples of every class, but it scores all C classes. This setting separates architecture compatibility from statistical coverage: a shared classifier is defined even when local class coverage is narrow.

2.2 Offline anchor construction

The offline stage constructs one anchor for each known class. A fixed prompt asks a language model for the visual characteristics of the class. It focuses on stable shape, texture, color, and object-level attributes and avoids dataset-specific backgrounds. A text-conditioned diffusion model renders a synthetic set $\widetilde{\mathcal{D}}_c$ from the description, and a frozen encoder ϕ embeds its images. The normalized

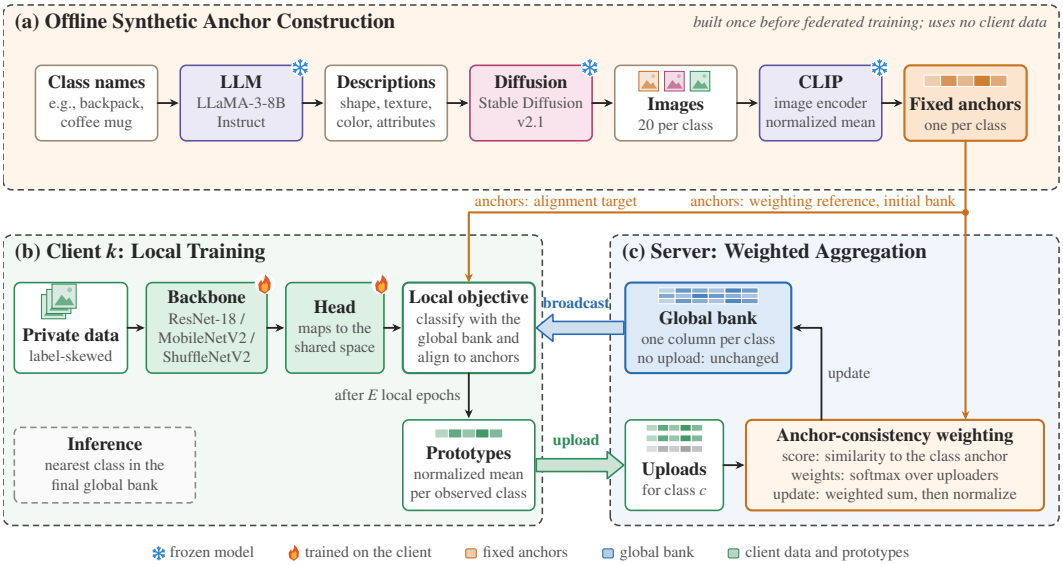

Figure 1: Overview of SATPL. (a) An LLM, a diffusion model, and a frozen CLIP encoder build one fixed anchor per class ($\mathcal{A}$) before training. (b) Each client trains its backbone and head against the received global bank $P^{(r)}$ and aligns its features to the anchors, then uploads class prototypes. (c) The server weights each upload by its similarity to the class anchor and forms the next bank $P^{(r+1)}$. Only class-indexed vectors are exchanged; private data, synthetic images, and model parameters are never transmitted.

mean defines the anchor:

$$\bar{\phi}_c = \frac{1}{|\widetilde{\mathcal{D}}_c|} \sum_{\tilde{x} \in \widetilde{\mathcal{D}}_c} \phi(\tilde{x}), \qquad A_c = \frac{\bar{\phi}_c}{\|\bar{\phi}_c\|_2}. \tag{1}$$

The bank $\mathcal{A} = \{A_c\}_{c=1}^{C}$ remains fixed during training. Its construction uses class-level descriptions rather than private examples. Clients receive only the anchor vectors; the synthetic images are not sent to them or mixed with their data.

Image anchors average many visual realizations of a description, so they can reflect variation in shape, texture, color, and viewpoint. A normalized CLIP text embedding of the description is a cheaper option with the same vector interface. Section 4.3 compares these constructions.

2.3 Local training with the global classifier

The server initializes a unit-norm prototype bank with the anchors, $P_c^{(0)} = A_c$. At round r, each active client receives $P^{(r)}$. For an example x, its normalized representation and class logits are

$$z_k(x) = g_k(f_k(x)), \qquad u_k(x) = \frac{z_k(x)}{\|z_k(x)\|_2}, \qquad s_{k,c}^{(r)}(x) = \frac{\langle u_k(x), P_c^{(r)} \rangle}{\mu}, \tag{2}$$

where $\mu > 0$ is the classifier temperature. The local objective combines cross-entropy and anchor alignment with weight $\lambda \geq 0$:

$$\mathcal{L}_k^{(r)} = \mathbb{E}_{(x,y)\sim\mathcal{D}_k} \left[-\log \frac{\exp(s_{k,y}^{(r)}(x))}{\sum_{c=1}^{C} \exp(s_{k,c}^{(r)}(x))} + \lambda\big(1 - \langle u_k(x), A_y \rangle\big) \right]. \tag{3}$$

Both $P^{(r)}$ and $\mathcal{A}$ remain constant during local optimization. Gradients update only f_k and g_k. The classifier term learns against the current client-derived bank. The alignment term pulls the representation toward a reference shared across rounds. The temperatures have separate roles: μ scales class logits, whereas the server temperature τ controls upload weights.

The bank shapes local gradients even though clients keep its vectors fixed. Let q_c be the softmax probability in Eq. (3), and suppress the client and round indices. Because $u = z/\|z\|_2$ has the Jacobian $(I - uu^\top)/\|z\|_2$, the chain rule gives, for one example,

$$\nabla_z \ell = \frac{I - uu^\top}{\|z\|_2} \left[\frac{1}{\mu} \left(\sum_{c=1}^{C} q_c P_c - P_y \right) - \lambda A_y \right]. \tag{4}$$

Changing an uploaded prototype can therefore change another client's gradient in the next round. A locally absent class still enters through its bank column in the softmax denominator. Each client thus learns to separate its own classes from all other classes in the federation.

2.4 Local prototypes and server aggregation

After training, a client extracts one prototype for each observed class. It normalizes features before averaging and then normalizes the class mean:

$$m_{k,c}^{(r)} = \frac{1}{|\mathcal{D}_{k,c}|} \sum_{(x,y)\in\mathcal{D}_{k,c}} u_k(x), \qquad p_{k,c}^{(r)} = \frac{m_{k,c}^{(r)}}{\|m_{k,c}^{(r)}\|_2}. \tag{5}$$

Feature normalization removes magnitude differences before the mean is formed. Centroid normalization puts each upload on the unit sphere. The number of class examples affects the estimate, but it is not a separate factor in the aggregation weight.

Let $\mathcal{K}_r$ be the active clients and $S_r(c)$ the clients uploading class c. For $k \in S_r(c)$, the server computes

$$\gamma_{k,c}^{(r)} = \frac{\exp(\langle p_{k,c}^{(r)}, A_c \rangle / \tau)}{\sum_{j\in S_r(c)} \exp(\langle p_{j,c}^{(r)}, A_c \rangle / \tau)}. \tag{6}$$

The softmax runs over the clients that upload class c in the current round. The server then forms and normalizes their weighted sum:

$$\widetilde{P}_c^{(r+1)} = \sum_{k\in S_r(c)} \gamma_{k,c}^{(r)} p_{k,c}^{(r)}, \qquad P_c^{(r+1)} = \frac{\widetilde{P}_c^{(r+1)}}{\|\widetilde{P}_c^{(r+1)}\|_2}. \tag{7}$$

If $S_r(c) = \emptyset$, the server retains $P_c^{(r+1)} = P_c^{(r)}$. This preserves a classifier column for every class.

Small τ concentrates weight on uploads closest to the anchor. As $\tau \to \infty$, weights approach uniform averaging within $S_r(c)$. A single upload always receives weight one, so weighting acts whenever several clients upload the same class. Similarity-only weighting keeps a client's influence independent of its class volume under label skew. A prototype estimated from very few samples can still receive a large weight if it happens to align closely, which makes this a deliberate design trade-off.

2.5 Prediction and information exchange

Prediction uses the final global bank and each client's last trained representation:

$$\hat{y}_k(x) = \operatorname*{arg\,max}_{c\in\{1,\dots,C\}} \langle u_k(x), P_c^{(T)} \rangle. \tag{8}$$

The positive temperature does not change the maximizing class, and each client scores all C classes. Clients that are inactive in the last round keep their parameters and also receive the final bank before evaluation.

The two references carry different information. Fixed anchors provide alignment targets and a server-side similarity reference. The global bank carries the participating clients' class information into later objectives. A fixed anchor classifier alone would not create this exchange when all model parameters remain local. Conversely, client-derived prototypes alone would provide exchange without an external reference for weighting. SATPL combines both.

An optional permissioned ledger can record anchor commitments and round-level aggregation metadata for provenance and post-hoc inspection. It works as an accountability layer alongside training. The learning signal and the robustness mechanism come from anchor alignment and anchor-consistency weighting.

Algorithm 1 SATPL training with fixed anchors and an evolving global classifier

Input: anchors $\mathcal{A}$; rounds T; local epochs E; weight λ; temperatures μ, τ
Output: global bank $P^{(T)}$ and each client's backbone f_k and projection head g_k

$P_c^{(0)} \leftarrow A_c$ for every class c; send $\mathcal{A}$ to all clients once
for $r = 0, 1, \ldots, T-1$ **do**
  Server selects the active clients $\mathcal{K}_r$ and broadcasts $P^{(r)}$
  for each client $k \in \mathcal{K}_r$ **in parallel do**
    Train f_k, g_k for E epochs on Eq. (3), keeping $P^{(r)}$ and $\mathcal{A}$ fixed
    Upload $p_{k,c}^{(r)}$ from Eq. (5) and its class index for every $c \in \mathcal{C}_k$
  end for
  for each class $c = 1, \ldots, C$ **do**
    if $S_r(c) \neq \emptyset$ **then**
      Compute $\gamma_{k,c}^{(r)}$ by Eq. (6) and $P_c^{(r+1)}$ by Eq. (7)
    else
      $P_c^{(r+1)} \leftarrow P_c^{(r)}$
    end if
  end for
end for
Each client predicts with its last f_k, g_k and the final bank $P^{(T)}$ by Eq. (8)

3 Model Integration and Analysis

3.1 A conditional bound on unreliable uploads

The weighting rule limits unreliable uploads when anchor similarity separates them from reliable uploads. Fix one class and round, and partition its uploaders into nonempty sets $\mathcal{G}$ and $\mathcal{B}$. These sets serve only the analysis, so the server needs no reliability labels. Write $a_k = \langle p_k, A \rangle$ and define

$$\Delta = \min_{i \in \mathcal{G}} a_i - \max_{j \in \mathcal{B}} a_j, \qquad w_{\mathcal{B}} = \sum_{j \in \mathcal{B}} \gamma_j. \tag{9}$$

A positive Δ means every reliable upload is more aligned with the anchor than every unreliable upload. The analysis is conditional on this gap.

Proposition 1. *For the weights in Eq.* (6)*,*

$$w_{\mathcal{B}} \leq \frac{|\mathcal{B}| e^{-\Delta/\tau}}{|\mathcal{G}| + |\mathcal{B}| e^{-\Delta/\tau}} \leq \frac{|\mathcal{B}|}{|\mathcal{G}|} e^{-\Delta/\tau}. \tag{10}$$

Let $\bar{p}_{\mathcal{G}}$ be the anchor-weighted mean formed using only reliable uploads. The unnormalized aggregate obeys $\|\widetilde{P} - \bar{p}_{\mathcal{G}}\|_2 \leq 2w_{\mathcal{B}}$. If $\|\bar{p}_{\mathcal{G}}\|_2 > 2w_{\mathcal{B}}$, the normalized aggregate is defined and satisfies

$$\left\| P - \frac{\bar{p}_{\mathcal{G}}}{\|\bar{p}_{\mathcal{G}}\|_2} \right\|_2 \leq \frac{4w_{\mathcal{B}}}{\|\bar{p}_{\mathcal{G}}\|_2}. \tag{11}$$

The proof bounds each reliable softmax numerator from below by $e^{\min_{\mathcal{G}} a/\tau}$ and each unreliable numerator from above by $e^{(\min_{\mathcal{G}} a - \Delta)/\tau}$. The direction bound follows from the convex decomposition $\widetilde{P} - \bar{p}_{\mathcal{G}} = w_{\mathcal{B}}(\bar{p}_{\mathcal{B}} - \bar{p}_{\mathcal{G}})$, where $\bar{p}_{\mathcal{B}}$ is the weighted mean of the unreliable uploads. For example, with equal group sizes and $\tau = 0.2$, margins of 0.2 and 0.4 limit the unreliable weight to about 0.27 and 0.12.

The norm condition reflects that the server broadcasts a direction. When the reliable uploads agree, their weighted mean has a large norm. Equation (11) then keeps the broadcast direction close to the reliable direction. The bound thus ties a stable update to two quantities: a small unreliable weight and agreement among reliable uploads.

3.2 Reading the bound

When $\Delta > 0$, the first bound in Eq. (10) lies below the unreliable group's uniform share. Anchor weighting then gives these uploads less influence than plain averaging would. Strict separation is

Table 1: Five-benchmark comparison: final accuracy (%), one fixed-seed run per setting. Best results are in bold.

Method	DomainNet	Office-Home	Ext. Yale B	COIL-100	Caltech-10
FedAvg-Head	35.40	58.20	86.10	82.30	80.50
FedProx-Head	37.12	60.45	87.50	84.10	81.80
FedProto	44.80	68.15	91.50	92.10	88.50
FedGen	43.20	65.90	90.50	90.10	86.10
FedLSA	52.10	73.40	95.20	94.50	92.30
SATPL	**56.75**	**76.90**	**97.40**	**96.23**	**94.60**

sufficient but not necessary. The unreliable group also receives less than its uniform share whenever its mean of $e^{a/\tau}$ is smaller than that of the reliable group. The proposition characterizes a single aggregation step, and Section 4.5 examines how the bank evolves over training.

The same anchor reference guides the alignment loss. It encourages different backbones to express each class near a common direction. A label-swapped upload is downweighted when it is dissimilar to the anchor of its claimed class. Both mechanisms rely on the anchor, so anchor quality and the similarity margin set the conditions under which they help. An upload that stays close to its claimed anchor is weighted like a reliable one.

SATPL also separates computational scale from architecture choice. If a round contains $M_r = \sum_c |S_r(c)|$ uploaded prototypes, server similarities and weighted sums cost $O(dM_r)$. The bank requires $O(dC)$ storage. Local training still depends on the backbone and dataset, as Section 4.7 shows.

4 Experimental Results and Analysis

4.1 Evaluation scope and protocol

The evaluation covers architecture heterogeneity, label skew, anchor construction, and corrupted prototype uploads. The five-benchmark study uses DomainNet, Office-Home, Extended Yale B, COIL-100, and Caltech-10 [12–16]. These benchmarks cover visual domains, illumination, viewpoints, and object categories. Anchor controls and recent baselines are run on DomainNet, Office-Home, and COIL-100, and matched seeds on DomainNet.

We report two bodies of evidence. The five-benchmark study gives fixed-seed summaries. Their run-level records and full image splits are not available, so Section 4.4 adds a fully logged Caltech-10 study with recorded configurations and per-client results. We read each study within its own protocol.

The benchmark protocol specifies 100 clients, ten active clients per round, 100 rounds, and five local epochs. Batch size is 64. SGD uses learning rate 0.01, momentum 0.9, weight decay 10^{-4}, and cosine decay. Label partitions use Dirichlet concentration 0.1. Clients receive ResNet-18, MobileNetV2, or ShuffleNetV2 backbones [17–19]. The shared dimension is 512, and $\mu = 0.1$. Hyperparameters are selected on a 10% local validation split from $\lambda \in \{0.5, 1, 1.5, 2\}$, $\tau \in \{0.1, 0.2, 0.5\}$, and $\mu \in \{0.05, 0.1, 0.2\}$. The selected (λ, τ) pairs are $(2.0, 0.2)$ for DomainNet and Office-Home, $(1.0, 0.5)$ for Extended Yale B, $(1.5, 0.1)$ for COIL-100, and $(1.5, 0.2)$ for Caltech-10. Test data play no part in tuning, baselines receive the same validation-based tuning budget, and we report final-round test accuracy.

The anchor pipeline uses LLaMA-3-8B-Instruct, Stable Diffusion v2.1, and CLIP ViT-B/32 [20–22]. Each description produces 20 synthetic images, and the anchor is the normalized mean of their 512-dimensional CLIP image embeddings.

4.2 Benchmark comparisons

SATPL achieves the highest accuracy on all five benchmarks in Table 1. It improves over FedLSA, the strongest baseline, by 4.65, 3.50, 2.20, 1.73, and 2.30 points, respectively. The comparison covers prototype exchange, generated-knowledge transfer, and learnable anchors. FedAvg-Head and FedProx-Head [1, 23] average only compatible shared layers, because the heterogeneous backbones

Table 2: Recent baselines (top) and anchor-construction controls of SATPL (bottom): accuracy (%). The anchor controls change only the anchor bank. Best results are in bold.

Method	DomainNet	Office-Home	COIL-100
FedType	52.8	74.1	92.6
FedTGP	54.4	75.0	93.9
FedSA	55.2	75.8	94.8
TinyProto	53.9	74.4	93.2
SATPL, random Gaussian anchors	51.2	73.5	89.2
SATPL, class-name CLIP text anchors	53.4	74.8	90.7
SATPL, description CLIP text anchors	54.1	75.3	92.4
SATPL, class name → image → CLIP	55.6	76.1	94.5
SATPL, description → image → CLIP (full)	**56.8**	**76.9**	**96.2**

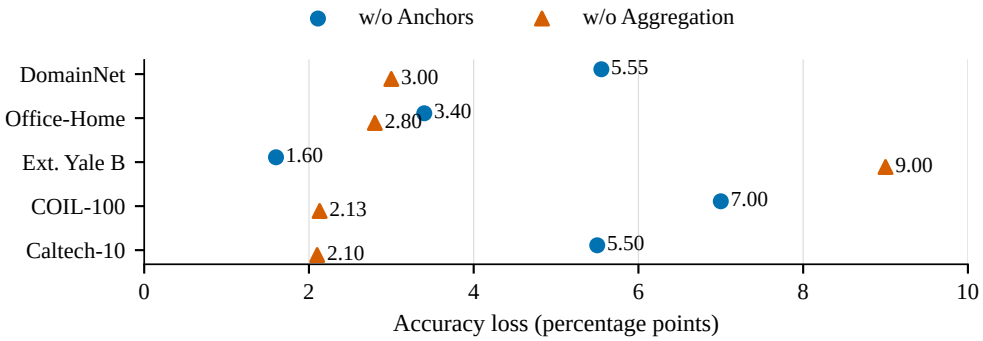

Figure 2: Component ablations on the five benchmarks. Each point is full SATPL accuracy minus the variant, in percentage points (fixed-seed summaries). w/o Anchors uses random anchors; w/o Aggregation uses ordinary averaging instead of anchor-consistency weighting.

cannot be averaged. Private linear adapters map MobileNetV2 and ShuffleNetV2 features to 512 dimensions, and ResNet-18 uses the identity. The server averages a shared 512-to-512 layer and the 512-to-C classifier. FedProx-Head adds a proximal term with coefficient 0.01 on the two shared weight matrices.

The top of Table 2 adds four recent methods: FedType, FedTGP, FedSA, and TinyProto. All methods share the same partitions, architecture assignments, participation schedules, and training budgets. SATPL ranks first on all three benchmarks. It leads FedSA, the strongest recent baseline, by 1.6, 1.1, and 1.4 points, so its advantage is not tied to one baseline family.

4.3 Anchor controls and component ablations

Two ablations isolate the two uses of the anchors. The w/o Anchors variant replaces the synthetic anchors with random anchors and keeps the training interface. The w/o Aggregation variant keeps the anchors, the global-bank classifier, and next-round feedback, but replaces anchor-consistency weighting with ordinary averaging. Figure 2 shows the accuracy each variant loses relative to full SATPL.

Both components contribute on every benchmark. Random anchors cost 5.55 points on DomainNet and 7.00 points on COIL-100. Ordinary averaging causes the largest drop on Extended Yale B, from 97.4% to 88.4%. The anchor controls at the bottom of Table 2 then change only the anchor bank, with partitions, architectures, participation, initialization, and training budget held fixed. Meaningful text anchors already outperform random anchors, and LLM descriptions improve them over bare class names. Image anchors add a further 2.7, 1.6, and 3.8 points over description text. With CLIP as the common interface, this comparison evaluates the image-generation branch as a whole. Text anchors remain a lower-cost alternative, while the full pipeline performs best.

Table 3: Logged Caltech-10 study after round 100: pooled local accuracy, mean ± sample SD over seeds 1–3 (%).

Method	Accuracy	SATPL variant or reference	Accuracy
SATPL	86.96 ± 2.86	Class-name text anchors	87.11 ± 1.60
FedProto	87.85 ± 1.12	Description text anchors	86.81 ± 2.72
FedAvg-Head	79.41 ± 2.19	Uniform aggregation	85.93 ± 3.03
FedProx-Head	79.56 ± 2.78	No alignment	85.19 ± 2.28
Local training	88.74 ± 2.00	Client-majority predictor	72.59 ± 0.71

Three matched seeds on DomainNet give $56.77 \pm 0.35\%$ for image anchors, $54.03 \pm 0.31\%$ for description text, and $55.20 \pm 0.30\%$ for FedSA (mean ± sample SD over seeds 42–44). Full SATPL leads FedSA in every matched run, by 1.57 points on average, and it also leads its text-anchor variant in every seed. Each SATPL variant keeps one fixed anchor bank across seeds, so the spread reflects federated training.

4.4 Caltech-10 with explicit client-level evaluation

The logged study selects the ten Office–Caltech categories from Caltech-256 [24]. It contains 1,123 images, with a fixed stratified test set of 225 images. The remaining 898 images are assigned to 100 clients by a client-wise Dirichlet scheme with concentration 0.1 and equal client sizes. Each client has eight or nine images, one of which it holds out for validation. The image list and partition may differ from the earlier Caltech-10 benchmark, so we compare methods within this study.

The three backbone families receive 34, 33, and 33 clients and start from ImageNet weights [25]. Clients use the same 100-round budget and the Caltech-10 settings above ($\lambda = 1.5$, $\tau = 0.2$, $\mu = 0.1$) without further tuning. With at most eight training images, each local epoch contains one gradient step. Seeds 1–3 vary the partition, architecture assignment, participation, and initialization, and comparisons within a seed share these choices.

Each method keeps its own classifier. SATPL and its controls use the final global bank with each client's last local representation. FedProto and local training use trainable local cosine classifiers, and FedProto also regularizes normalized features toward count-weighted global prototypes. The two head-averaging baselines use the final global shared layers with private backbones and adapters. If clients keep their last local head copies instead, these baselines reach 81.6% and 81.9% pooled local accuracy.

The metric depends on which client classifies each test image. Let $\mathcal{T}$ be the test set and $o(x)$ the client assigned to image x, drawn in proportion to each client's training count for the image's class. We use pooled local accuracy as the main metric and also report full-test accuracy:

$$\mathrm{Acc}_{\mathrm{local}} = \frac{1}{|\mathcal{T}|} \sum_{x \in \mathcal{T}} \mathbf{1}[\hat{y}_{o(x)}(x) = y(x)], \tag{12}$$

$$\mathrm{Acc}_{\mathrm{all}} = \frac{1}{K|\mathcal{T}|} \sum_{k=1}^{K} \sum_{x \in \mathcal{T}} \mathbf{1}[\hat{y}_k(x) = y(x)]. \tag{13}$$

Pooled local accuracy measures how well each client serves its own data distribution. Full-test accuracy gives every client the same images, including classes it never observed.

SATPL improves pooled local accuracy over the two head-averaging baselines by about 7.5 points. It does so while exchanging about 1.2% of their per-round bytes (Section 4.7). FedProto and local training, which keep trainable local classifiers, score 0.9 and 1.8 points higher on this small subset. Class-name and description text anchors stay within 0.2 points of image anchors here. The client-majority predictor reaches 72.6%, which shows how strongly the local label skew shapes this metric.

Full-test accuracy asks every client to recognize all ten classes, although a client holds about two of them. All methods therefore stay low on this metric. SATPL reaches 15.8%, FedProto 16.9%, local training 17.6%, and the head-averaging baselines about 20%. The prototype and cosine classifiers score zero on classes that are absent from a client's training data, whereas the averaged linear heads reach about 6%.

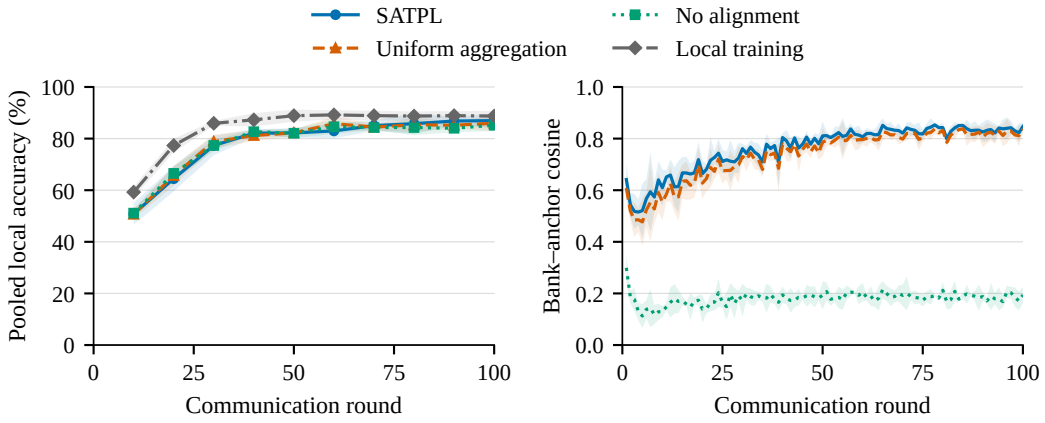

Figure 3: Logged Caltech-10 dynamics, seeds 1–3. Left: pooled local accuracy at the ten evaluation rounds. Right: mean cosine similarity between bank columns and their class anchors, recorded each round. Lines show means; shading shows ± 1 sample SD. Straight segments join the measured rounds.

4.5 Learning dynamics in the logged study

Figure 3 uses recorded rounds from the Caltech-10 study. Accuracy is evaluated every ten rounds, and bank similarity is logged after every round. The three seeds share the same test images, but their client assignments change with the training partition. The bands therefore combine training variability with changes in which client predicts each image.

Alignment shapes the bank geometry strongly. With alignment, the final mean bank–anchor cosine reaches about 0.85, against 0.19 without it, and uniform aggregation ends near 0.84. Both components also improve accuracy. Removing alignment lowers mean pooled local accuracy by 1.8 points, and removing anchor weighting lowers it by 1.0 point. Neither removal helps in any seed.

4.6 Gaussian prototype corruption

The corruption study selects a fixed share $\rho \in \{0.2, 0.4\}$ of clients for each seed. Whenever a selected client uploads, each prototype becomes $q = (p + \xi)/\|p + \xi\|_2$ with fresh $\xi \sim \mathcal{N}(0, I_d/d)$. The expected squared perturbation norm thus equals the squared norm of the clean upload. Figure 4 compares anchor weighting with uniform aggregation using paired seeds.

Anchor weighting raises bank–anchor similarity in all nine paired conditions, and the gap grows with the corruption share. The weighted bank therefore stays closer to the anchors when uploads are corrupted, which is consistent with the mechanism behind Proposition 1. The mean accuracy advantage is 1.0 point without corruption and stays positive at both corruption shares, at 0.7 and 0.1 points, with seed-level differences of both signs.

4.7 Computation and communication

The message sizes follow directly from the shared interface. An active client uploads $4d|\mathcal{C}_k|$ FP32 bytes plus class indices and downloads $4dC$ bytes for the bank. With $d = 512$ and ten held classes, the vector upload is 20,480 bytes, and a 345-class DomainNet bank occupies 706,560 bytes.

In the logged Caltech-10 setting, prototypes and int32 class indices total about 25.1 kB per active client and round. Head averaging exchanges $8(d^2 + d + dC + C)$ bytes in both directions, about 2.14 MB, so SATPL needs about 1.2% of that traffic. Communication thus scales with the class count and the prototype dimension rather than with backbone size.

The computation stays modest as well. A matched DomainNet timing comparison gives stable rounds of 33.15 seconds for FedProto, 36.45 seconds for SATPL, 38.62 seconds for FedSA, and 43.80 seconds for FedTGP. The SATPL server stage takes 0.98 seconds, about 2.7% of its round,

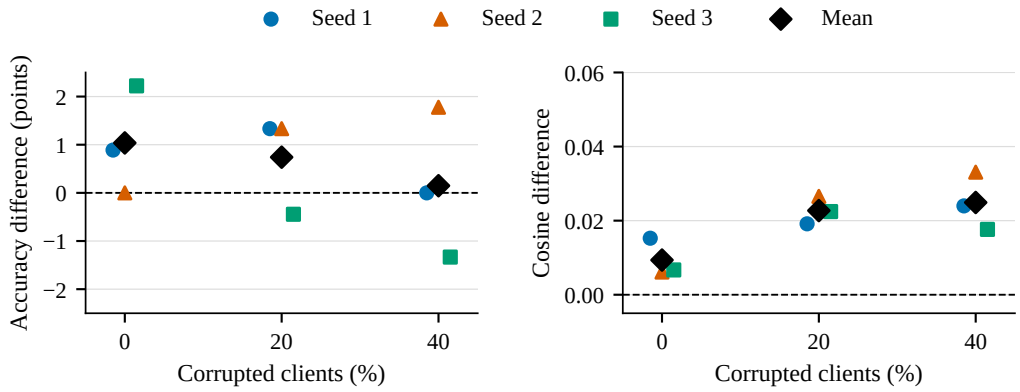

Figure 4: Logged Caltech-10 corruption study. Each point is SATPL minus uniform aggregation for one paired seed; black diamonds show the three-seed mean. Left: pooled local accuracy difference. Right: final bank–anchor cosine difference. The zero-corruption runs are those of Table 3; fresh Gaussian noise is applied at 20% and 40% corrupted clients.

and similarity-weighted aggregation itself takes 0.82 seconds. In the logged Caltech-10 study, local training takes about 4.4 seconds and prototype extraction about 0.5 seconds per active-client update. Aggregation takes about three milliseconds per round, and the one-time anchor construction for the ten classes takes 18.6 minutes.

4.8 Practical considerations

SATPL works best when class names carry visual meaning for the external models, as object categories do. Identity or instance labels, such as faces or individual objects, need descriptions that capture their distinguishing attributes. Generic descriptions can also miss rare subgroups or highly stylized domains, which the multi-anchor extension in Section 5 targets.

SATPL assumes a fixed shared class vocabulary and a common embedding dimension, while it allows different backbone architectures. Adding a class means adding an anchor and a bank column. Prototype sharing keeps raw examples on the clients, but class prototypes can still reveal which classes a client holds. Formal privacy mechanisms such as differential privacy are complementary to this design. The external models bring a one-time compute cost and their own biases. Deployments should also state whether their accuracy target is each client's local distribution or the full class set.

5 Conclusion

SATPL gives heterogeneous clients a shared semantic coordinate system built from two references. Fixed synthetic anchors come from class descriptions in one offline run. They give every client the same alignment target and give the server a signal for weighting uploads. The global prototype bank changes every round and serves as each client's classifier, so other clients' prototypes shape every local loss. Clients with different backbones thus exchange only class-indexed vectors. Proposition 1 states when anchor-consistency weighting suppresses unreliable uploads and bounds the resulting shift in the broadcast direction.

The experiments support this design. In the five-benchmark summaries, SATPL ranks first, leads four recent baselines, and keeps its lead over FedSA in every matched DomainNet seed. The anchor controls favor the full image pipeline, and the ablations show that both the anchors and anchor-consistency weighting contribute. In the fully logged Caltech-10 study, alignment and anchor weighting each raise mean pooled local accuracy, and anchor weighting keeps the bank anchor-consistent under corrupted uploads. There, SATPL exceeds head averaging by 7.5 points with about 1% of its message size, while FedProto and local training stay slightly ahead on this small subset. Multi-anchor classes, with one generic anchor and several style-conditioned anchors, are a natural next step for stylized domains, together with larger logged benchmarks.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 62506016), the 2026 Beijing Education Science Planning Project of the 15th Five-Year Plan (Grant No. ACGA26138), and the Science and Technology Development Fund, Macao S.A.R. (FDCT) (Grant No. 0028/2023/RIA1).

The authors declare no competing interests.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017. URL `https://proceedings.mlr.press/v54/mcmahan17a.html`.

[2] J. Zhang, X. Wu, Y. Zhou, X. Sun, Q. Cai, Y. Liu, Y. Hua, Z. Zheng, J. Cao, and Q. Yang. HtFLlib: A comprehensive heterogeneous federated learning library and benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 5900–5911. ACM, 2025. doi: 10.1145/3711896.3737379.

[3] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang. FedProto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8432–8440, 2022. doi: 10.1609/aaai.v36i8.20819.

[4] Z. Zhu, J. Hong, and J. Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12878–12889. PMLR, 2021. URL `https://proceedings.mlr.press/v139/zhu21b.html`.

[5] L. Fu, S. Huang, Y. Lai, T. Liao, C. Zhang, and C. Chen. Beyond federated prototype learning: Learnable semantic anchors with hyperspherical contrast for domain-skewed data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16648–16656, 2025. doi: 10.1609/aaai.v39i16.33829.

[6] Y. Zhou, X. Qu, C. You, J. Zhou, J. Tang, X. Zheng, C. Cai, and Y. 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, pages 23009–23017, 2025. doi: 10.1609/aaai.v39i21.34464.

[7] J. Wang, C. Zhao, L. Lyu, Q. You, M. Huai, and F. Ma. Bridging model heterogeneity in federated learning via uncertainty-based asymmetrical reciprocity learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 52290–52308. PMLR, 2024. URL `https://proceedings.mlr.press/v235/wang24cs.html`.

[8] J. Zhang, Y. Liu, Y. Hua, and J. 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, pages 16768–16776, 2024. doi: 10.1609/aaai.v38i15.29617.

[9] G. Lee and D. Choi. Communication-efficient heterogeneous federated learning with sparse prototypes in resource-constrained environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 22787–22795, 2026. doi: 10.1609/aaai.v40i27.39441.

[10] Z. Hao, H. Li, B. Zhang, and J. Du. FRIL: Blockchain-powered heterogeneous model fusion for robust food recognition. *IEEE Transactions on Multimedia*, pages 1–14, 2026. doi: 10.1109/TMM.2026.3721338.

[11] Z. Hao, L. Cao, H. Yu, X. Teng, G. Wang, and C. Zhang. LLM-powered data synthesis on blockchain for fusion isomerism learning in heterogeneous federated systems. *CAAI Transactions on Intelligence Technology*, 2026. doi: 10.1049/cit2.70186.

[12] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019. doi: 10.1109/ICCV.2019.00149.

[13] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Venkateswara_Deep_Hashing_Network_CVPR_2017_paper.html.

[14] K.-C. Lee, J. Ho, and D. J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5): 684–698, 2005. doi: 10.1109/TPAMI.2005.92.

[15] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-100). Technical Report CUCS-006-96, Columbia University, February 1996. URL https://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php.

[16] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2012. doi: 10.1109/CVPR.2012.6247911.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. doi: 10.1109/CVPR.2018.00474.

[19] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *Proceedings of the European Conference on Computer Vision*, pages 116–131, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Ningning_Light-weight_CNN_Architecture_ECCV_2018_paper.html.

[20] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783.

[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

[23] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html.

[24] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report CNS-TR-2007-001, California Institute of Technology, 2007. URL https://authors.library.caltech.edu/records/5sv1j-ytw97.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

A Proofs and the Local Gradient

A.1 Weight assigned to unreliable uploads

Fix a class and round as in Proposition 1. All uploads and the anchor have unit norm, and $\tau > 0$. The sets $\mathcal{G}$ and $\mathcal{B}$ are nonempty. Define

$$Z_{\mathcal{G}} = \sum_{i \in \mathcal{G}} e^{a_i/\tau}, \quad Z_{\mathcal{B}} = \sum_{j \in \mathcal{B}} e^{a_j/\tau}, \quad a_{\min} = \min_{i \in \mathcal{G}} a_i, \quad b_{\max} = \max_{j \in \mathcal{B}} a_j. \tag{14}$$

The group sums satisfy $Z_{\mathcal{G}} \geq |\mathcal{G}| e^{a_{\min}/\tau}$ and $Z_{\mathcal{B}} \leq |\mathcal{B}| e^{b_{\max}/\tau}$. Since $w_{\mathcal{B}} = Z_{\mathcal{B}}/(Z_{\mathcal{G}} + Z_{\mathcal{B}})$, monotonicity in the two sums gives

$$w_{\mathcal{B}} \leq \frac{|\mathcal{B}| e^{b_{\max}/\tau}}{|\mathcal{G}| e^{a_{\min}/\tau} + |\mathcal{B}| e^{b_{\max}/\tau}} \tag{15}$$

$$= \frac{|\mathcal{B}| e^{-\Delta/\tau}}{|\mathcal{G}| + |\mathcal{B}| e^{-\Delta/\tau}} \leq \frac{|\mathcal{B}|}{|\mathcal{G}|} e^{-\Delta/\tau}. \tag{16}$$

This proves Eq. (10). The first expression remains bounded by one even when the simpler second expression is loose.

Define the within-group weighted means

$$\bar{p}_{\mathcal{G}} = \sum_{i \in \mathcal{G}} \frac{e^{a_i/\tau}}{Z_{\mathcal{G}}} p_i, \qquad \bar{p}_{\mathcal{B}} = \sum_{j \in \mathcal{B}} \frac{e^{a_j/\tau}}{Z_{\mathcal{B}}} p_j. \tag{17}$$

Both means have norm at most one because they are convex combinations of unit vectors. The full aggregate is

$$\widetilde{P} = (1 - w_{\mathcal{B}}) \bar{p}_{\mathcal{G}} + w_{\mathcal{B}} \bar{p}_{\mathcal{B}}, \qquad \|\widetilde{P} - \bar{p}_{\mathcal{G}}\|_2 \leq w_{\mathcal{B}} (\|\bar{p}_{\mathcal{B}}\|_2 + \|\bar{p}_{\mathcal{G}}\|_2) \leq 2 w_{\mathcal{B}}. \tag{18}$$

For any nonzero vectors x, y, the triangle and reverse triangle inequalities imply

$$\left\| \frac{x}{\|x\|_2} - \frac{y}{\|y\|_2} \right\|_2 \leq \frac{|\|y\|_2 - \|x\|_2| + \|x - y\|_2}{\|y\|_2} \leq \frac{2\|x - y\|_2}{\|y\|_2}. \tag{19}$$

If $\|\bar{p}_{\mathcal{G}}\|_2 > 2 w_{\mathcal{B}}$, Eq. (18) ensures that $\widetilde{P} \neq 0$. Substituting $x = \widetilde{P}$ and $y = \bar{p}_{\mathcal{G}}$ into Eq. (19) proves Eq. (11).

The strict margin has a precise relation to uniform weighting. Let $\bar{e}_{\mathcal{G}} = Z_{\mathcal{G}}/|\mathcal{G}|$ and $\bar{e}_{\mathcal{B}} = Z_{\mathcal{B}}/|\mathcal{B}|$. Direct cross-multiplication gives

$$w_{\mathcal{B}} < \frac{|\mathcal{B}|}{|\mathcal{G}| + |\mathcal{B}|} \quad \Longleftrightarrow \quad \bar{e}_{\mathcal{B}} < \bar{e}_{\mathcal{G}}. \tag{20}$$

A positive minimum gap implies this inequality, but overlapping similarity ranges can also satisfy it. If $\mathcal{B}$ is empty, its weight is zero. If $\mathcal{G}$ is empty, no reliable-only reference exists, so the direction comparison does not apply.

A.2 How the global bank changes local optimization

For one example with label y, write $s_c = u^\top P_c/\mu$ and $q_c = e^{s_c}/\sum_j e^{s_j}$. Differentiating cross-entropy with respect to s_c gives $q_c - \mathbf{1}[c = y]$. Since the bank and anchors are fixed during the local update,

$$\nabla_u \ell = \frac{1}{\mu} \left(\sum_{c=1}^{C} q_c P_c - P_y \right) - \lambda A_y. \tag{21}$$

For $z \neq 0$ and $u = z/\|z\|_2$, the normalization Jacobian is $(I - uu^\top)/\|z\|_2$. The chain rule then gives Eq. (4). Gradients reach the backbone and projection through their Jacobians with respect to z. An upload influences this expression through the next bank, both directly through its vectors and indirectly through q_c.

This derivative identifies the exchange mechanism. The numerical implementation uses a norm floor near zero, and the displayed derivative describes the nondegenerate region.

B Experimental Protocols and Implementation

B.1 Settings of the five-benchmark study

Table 4 lists the five-benchmark settings. The search uses a 10% local validation split and fixes the selected parameters for testing. The candidates are $\lambda \in \{0.5, 1, 1.5, 2\}$, $\tau \in \{0.1, 0.2, 0.5\}$, and $\mu \in \{0.05, 0.1, 0.2\}$. The selected classifier temperature is 0.1 for every dataset.

Table 4: Five-benchmark settings. The common optimization settings appear in Section 4.1.

Dataset	λ	τ
DomainNet	2.0	0.2
Office-Home	2.0	0.2
Extended Yale B	1.0	0.5
COIL-100	1.5	0.1
Caltech-10	1.5	0.2

The benchmarks use class categories, face identities, or object instances as their labels. DomainNet and Office-Home also contain several visual domains, and the summaries report dataset-level accuracy.

B.2 Logged Caltech-10 data and models

The logged dataset comes from the Caltech-256 archive. Table 5 gives the complete category mapping. The fixed test split uses seed 20261013 and contains 225 images. The remaining 898 images enter the client partition. Each client holds out one image for validation. This produces 798 training images across 100 clients.

Table 5: Logged Caltech-10 class mapping. The folder suffix identifies the category in the Caltech-256 archive.

Folder suffix	Anchor class name
`backpack`	backpack
`touring-bike`	bicycle
`calculator`	calculator
`head-phones`	headphones
`computer-keyboard`	computer keyboard
`laptop-101`	laptop computer
`computer-monitor`	computer monitor
`computer-mouse`	computer mouse
`coffee-mug`	coffee mug
`video-projector`	video projector

Images are scaled by their shorter side and center-cropped to 224×224. The models use ImageNet channel normalization. Training adds a random crop after padding by 28 pixels and a horizontal flip. The backbone assignment uses ResNet-18, MobileNetV2, and ShuffleNetV2 $\times 1.0$, with torchvision `IMAGENET1K_V1` weights. Their pooled features feed FC(512)–ReLU–FC(512) projection heads. The learning rate decays over rounds, and each client retains its momentum buffers between rounds.

Batch-normalization statistics are re-estimated after local training for every method. Each client's data fit in one batch. The implementation therefore computes 50 moving-average updates from one forward pass in closed form. It clamps the stationary variance estimate at zero and floors running variances at 10^{-10}. These numerical details belong to the recorded Caltech-10 implementation.

The main methods, baseline methods, and corrupted-upload runs use the same seed-dependent partitions, architecture assignments, and participation schedules. Table 6 gives the pooled local scores before rounding their means. Sample standard deviations use denominator $n - 1$, with $n = 3$. They describe training-seed variation conditional on one image split and one anchor bank per variant.

B.3 Anchor construction and numerical safeguards

The prompt requests stable class-level visual attributes:

Table 6: Logged Caltech-10 pooled local accuracy by seed (%). Each client predicts using its last local representation after the 100-round schedule.

Method	Seed 1	Seed 2	Seed 3
SATPL	90.22	84.89	85.78
FedProto	88.00	86.67	88.89
FedAvg-Head	76.89	80.89	80.44
FedProx-Head	76.44	81.78	80.44
Class-name text anchors	88.44	85.33	87.56
Description text anchors	89.78	84.44	86.22
Uniform aggregation	89.33	84.89	83.56
No alignment	87.11	82.67	85.78
Local training	90.67	86.67	88.89
Client-majority predictor	71.78	72.89	73.11

> Describe the visual characteristics of the class [c] for image classification. Focus on stable shape, texture, color, and object-level attributes. Avoid dataset-specific background details.

The class name replaces `[c]`. In the Caltech-10 construction, LLaMA-3-8B-Instruct uses bfloat16, temperature 0.7, top-p 0.9, top-k 50, and at most 160 new tokens. The description seed is 20261013 plus the class index. Descriptions are truncated to the text encoders' 77-token context.

Stable Diffusion v2.1 uses FP16 weights, a DDIM scheduler, 50 denoising steps, guidance scale 7.5, and no negative prompt. It generates 20 images per class at 768×768 pixels. The image seed is $20261013 + 100000v + 1000c + i$, where $v = 1$ for description images, $c \in \{1, \dots, 10\}$, and $i \in \{0, \dots, 19\}$. The class order follows Table 5. CLIP ViT-B/32 supplies raw projected image features. Equation (1) averages these raw features and then normalizes the mean. The text controls instead normalize one projected text embedding of the class name or description. Each bank is fixed across seeds 1–3.

Prototype extraction uses inference mode without training augmentation. It excludes nonfinite representations and representations with norm below 10^{-8}. It normalizes each valid feature, averages within class, and normalizes the mean. A nonfinite or near-zero mean produces no valid upload. The server subtracts the largest scaled similarity before evaluating the softmax. This changes no exact weight but avoids exponential overflow. If a weighted aggregate is nonfinite or has norm below 10^{-8}, the Caltech-10 implementation keeps the previous class vector. The theory concerns nondegenerate aggregates.

B.4 Baseline interfaces and test classifiers

The head-averaging baselines use compatible shared parameters. A private linear adapter maps MobileNetV2 features from 1,280 to 512 dimensions and ShuffleNetV2 features from 1,024 to 512. ResNet-18 uses the identity adapter. The shared head is FC(512)–ReLU–FC(C), including biases. The backbones, adapters, batch-normalization states, and optimizer buffers remain local. In the Caltech-10 implementation, server averaging weights each client by its number of training examples.

FedProx-Head adds a proximal penalty with coefficient 0.01 on the two shared weight matrices. Its biases are excluded. The main evaluation uses the final global shared head with each client's last local representation. If each client instead retains its own last head, mean pooled local accuracy becomes 81.6% for FedAvg-Head and 81.9% for FedProx-Head.

The logged FedProto baseline uses a trainable local cosine classifier with $\mu = 0.1$. It adds $\|u_k(x) - \bar{q}_y\|_2^2$ with coefficient one, where $\bar{q}_y$ is the count-weighted, unnormalized global prototype from the preceding round. A class absent from that round has no regularization target. This normalized-feature implementation differs in scale from the original FedProto formulation on unnormalized features. Local training uses the same local cosine classifier without prototype exchange. SATPL and its anchor and aggregation controls use the final global bank in Eq. (8).

The majority predictor uses only each client's training label counts. For a test image assigned to client k, it predicts uniformly among classes attaining the largest count. Its reported accuracy averages the

probability of a correct prediction under ties. Fractional correct counts are therefore expected values. Every classifier scores all classes at test time.

C Matched-Seed Results

Table 7 lists the matched-seed values on DomainNet, the benchmark with the largest class set. Each SATPL variant uses one anchor bank, generated once before training and reused for seeds 42–44. Within each seed, the partition, architecture assignment, participation sequence, and initialization protocol are matched across methods. The summary gives the mean and sample standard deviation of the three runs, so the spread reflects federated training with each anchor bank held fixed.

Table 7: Reported DomainNet accuracy for matched seeds (%). The summary is mean ± sample SD over the three listed values.

Method	Seed 42	Seed 43	Seed 44	Summary
FedSA	55.2	54.9	55.5	55.20 ± 0.30
SATPL, description text	54.1	53.7	54.3	54.03 ± 0.31
SATPL, description images	56.8	56.4	57.1	56.77 ± 0.35

D Compute and Communication Details

D.1 DomainNet timing

In the matched timing comparison, all methods share 100 clients with ten active per round, 100 rounds, five local epochs, batch size 64, and the same partition, architecture assignment, participation sequence, and seed. Wall-clock times use `time.perf_counter()` with all GPUs synchronized at stage boundaries. Table 8 reports stable-round statistics over rounds 10–100. Server time covers inter-process collection and coordination and server aggregation or optimization.

Table 8: DomainNet timing in seconds. Stable-round values are mean ± spread over rounds 10–100.

Method	Stable round	Server per round
FedProto	33.15 ± 0.42	0.08
SATPL	36.45 ± 0.48	0.98
FedSA	38.62 ± 0.55	0.54
FedTGP	43.80 ± 0.51	5.42

The SATPL stable round contains 33.24 seconds of local training, 1.05 seconds of local prototype extraction, 0.16 seconds of prototype collection and coordination, 0.82 seconds of similarity-weighted aggregation and normalization, and 1.18 seconds of synchronization and other overhead. These components sum to 36.45 seconds. The server stage therefore takes 0.98 seconds, about 2.7% of the round; similarity-weighted aggregation itself takes 0.82 seconds, about 2.2%.

D.2 Caltech-10 compute and payload

The one-time anchor construction for the ten classes takes 18.6 minutes, including model loading and I/O. Its main components are 94 seconds for descriptions, 996 seconds for image generation, and seven seconds for CLIP encoding.

An uploaded prototype has 512 FP32 values and one int32 class index, totaling 2,052 bytes. Active clients hold about 2.24 classes on average, giving about 4,596 uploaded bytes per client and round. The ten-class bank adds 20,480 downloaded bytes. A head-averaging baseline instead exchanges 267,786 parameters in each direction: $512^2 + 512 + 512 \times 10 + 10$. This gives 2,142,288 bytes per round in both directions, and the 1.2% payload ratio follows from these arrays.

The per-update training and extraction times in Section 4.7 are means over active-client operations. Server aggregation times are per round.

NeurIPS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: The abstract and Section 1 distinguish the method, conditional theory, and two empirical scopes.
2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: Sections 3.2, 4.1, 4.4, and 4.8 discuss assumptions, metric dependence, evidence scope, and privacy.
3. **Theory assumptions and proofs**
   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
   Answer: [Yes]
   Justification: Proposition 1 and Appendix A supply assumptions, proofs, and the gradient derivation.
4. **Experimental result reproducibility**
   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?
   Answer: [No]
   Justification: The recorded study is fully specified (Appendix B); the five-benchmark image splits and run records are not available.
5. **Open access to data and code**
   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?
   Answer: [No]
   Justification: The implementation is not released with the paper; Appendix B gives the settings.
6. **Experimental setting/details**
   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?
   Answer: [No]
   Justification: Section 4.1 and Appendix B give the training and test settings; the five-benchmark partitions are not available.
7. **Experiment statistical significance**
   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?
   Answer: [No]
   Justification: Repeated-run uncertainty is available only for DomainNet and recorded Caltech-10 results (Appendix C, Section 4.4).
8. **Experiments compute resources**
   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
   Answer: [No]
   Justification: Appendix D gives per-round and offline timings; total compute across all runs is not reported.
9. **Code of ethics**
   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
   Answer: [Yes]
   Justification: The study uses existing benchmarks, collects no new human data, and discloses privacy and evidence limits (Section 4).

10. **Broader impacts**
    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
    Answer: [Yes]
    Justification: Sections 2.5 and 4.8 address data retention, information exposure, external-model bias, and adversarial limits.
11. **Safeguards**
    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?
    Answer: [N/A]
    Justification: No high-risk pretrained model, new image collection, or scraped dataset is released.
12. **Licenses for existing assets**
    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?
    Answer: [No]
    Justification: Citations credit the assets; version and license records are incomplete for the earlier runs.
13. **New assets**
    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?
    Answer: [N/A]
    Justification: No new public dataset or pretrained model is released.
14. **Crowdsourcing and research with human subjects**
    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?
    Answer: [N/A]
    Justification: No crowdsourcing or participant recruitment is reported.
15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**
    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?
    Answer: [N/A]
    Justification: No new human-participant experiment is reported.
16. **Declaration of LLM usage**
    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.
    Answer: [Yes]
    Justification: Section 2.2 and Appendix B.3 describe the LLM, prompt, generation, and truncation.