Agent-Orchestrated Federated Representation Learning for Non-IID Data

Lei Yao
Department of Computer Science
University of Wisconsin – Milwaukee
Milwaukee, WI, USA
leiyao@uwm.edu

Tian Zhao
Department of Computer Science
University of Wisconsin – Milwaukee
Milwaukee, WI, USA
tzhao@uwm.edu

***Abstract*—Training high-fidelity, domain-specific models demands large-scale, high-quality datasets; however, stringent privacy requirements often create significant barriers to data sharing in sensitive sectors. While traditional Federated Learning (FL) addresses these concerns by exchanging model gradients rather than raw data, its efficacy is frequently compromised by suboptimal convergence and the challenges of managing non-IID data. To tackle these challenges, we present AO-FRL (Agent-Orchestrated Federated Representation Learning), a novel framework that addresses privacy preservation and data heterogeneity in collaborative learning. Instead of averaging model parameters, AO-FRL aggregates privacy-protected feature representations extracted by client-side encoders. These representations are norm-clipped and perturbed with calibrated Gaussian noise to enable differentially private representation exchange. Through multi-agent communication, the server agent further orchestrates client uploads via dynamic per-class budget allocation, improving class coverage and training coordination under non-IID client heterogeneity.**

On CIFAR-100 with differentially private noise $\sigma = 0.02$, AO-FRL achieves 66.66% accuracy, outperforming the Fed-series baselines trained with 10 local epochs per round, including FedAvg by 7.57% and FedProx by 8.67%, while reaching its best accuracy within only 8 rounds. Under the same noise-free setting, AO-FRL converges with only 1,132 MB of communication, compared with 2,017 MB for FedAdam. In terms of wall-clock time, noise-free AO-FRL finishes in 2m33s, about $5.9\times$ faster than FedAvg and $9.2\times$ faster than FedProx. The server agent's dynamic per-class upload-budget allocation further improves long-tail learning under non-IID distributions, with the most improved class increasing from 33.6% to 55.2%. Overall, AO-FRL demonstrates that multi-agent collaboration enables automated training and dynamic adaptation, allowing the system to adjust resource allocation in response to heterogeneous data conditions and achieve stable, effective learning in privacy-sensitive domains.

***Index Terms*—Agents Collaboration, Federated Learning, Adaptive Orchestration, Privacy-Preserving Training**

I. Introduction

The rapid development of machine learning has enabled powerful models to be trained on increasingly large and diverse datasets. However, in many real-world applications, such as healthcare, finance, and personalized services, data is inherently distributed across multiple organizations or devices and cannot be centrally aggregated due to privacy, security, and regulatory constraints. This limitation has motivated the emergence of Federated Learning (FL) [1], a paradigm that enables collaborative model training without requiring raw data to leave local environments.

Despite its privacy advantages, traditional federated learning based on parameter averaging (e.g. FedAvg) faces significant challenges in practice. In realistic scenarios, client data is often non-IID (non-independent and identically distributed), exhibiting variations in class distribution, sample size, and feature characteristics. Under such heterogeneity, gradient-based aggregation tends to suffer from slow convergence, model bias toward majority clients, and substantial performance degradation [2]. These issues are particularly pronounced in domains with severe data imbalance, where minority classes receive insufficient representation in the global model.

To address these limitations, recent studies have explored alternative paradigms such as representation-level collaboration and synthetic data generation. However, directly sharing representations or synthetic samples introduces new privacy risks, as embeddings or generated data may still contain sensitive information about local datasets. Moreover, existing approaches often rely on manually configured training pipelines, lacking the ability to dynamically adapt to evolving data conditions across clients.

In this paper, we propose AO-FRL (Agent-Orchestrated Federated Representation Learning), a novel framework that rethinks federated learning from a multi-agent perspective. Rather than aggregating model parameters as in conventional federated learning, AO-FRL operates at the data level by aggregating privacy-protected representations of client data for centralized training. By treating the server and clients as autonomous agents, our framework enables continuous Agent-to-Agent (A2A) communication, allowing the training process to be dynamically orchestrated according to data heterogeneity and privacy constraints. Through this collaborative mechanism, the server adaptively adjusts client upload budgets, ensuring balanced representation across classes while maintaining automated and stable learning.

To further mitigate privacy risks, AO-FRL incorporates a privacy-aware representation sharing mechanism. Each client converts local data into feature embeddings, applies L2-norm clipping to bound sensitivity, and adds Gaussian noise calibrated to representation sensitivity under the differential privacy framework [3]. This design provides practical privacy

protection while preserving useful information for global training, as further verified by our decoder-based reconstruction evaluation using PSNR (peak signal-to-noise ratio). Unlike most traditional federated learning algorithms that transmit model parameters, AO-FRL performs training on the server by sharing noise-perturbed embeddings, thereby significantly reducing the computational and optimization burden on clients. Figure 1 shows the overall workflow of AO-FRL.

This paper makes three primary contributions:

1) Novel Framework: AO-FRL is a FL-inspired framework that uses multi-agent communication to automate server–client coordination without manual intervention. Instead of aggregating locally trained parameters, AO-FRL collects privacy-protected representations from non-IID clients and consolidates them on the server to form a more balanced training set. This design enhances privacy, reduces distributional imbalance, and shifts training to the server, thereby lowering client-side hardware requirements.
2) Adaptive Orchestration for Long-Tail Fairness: AO-FRL introduces a server-side dynamic budget allocation strategy that adaptively assigns per-class upload budgets based on data heterogeneity and class-specific learning difficulty. Without manual tuning, it redirects communication toward under-represented and hard-to-learn classes, improving long-tail fairness under non-IID distributions. On CIFAR-100, AO-FRL improves the mean accuracy of the 10 worst classes by 11% and raises the worst-class accuracy from 4% to 29%.
3) Comprehensive Empirical Validation: Extensive experiments on CIFAR-100, CIFAR-10, and SVHN under severe heterogeneity demonstrate the effectiveness of AO-FRL. With a Dirichlet partition of $\alpha = 0.3$ and 20 clients, AO-FRL outperforms FedAvg, FedProx, and FedAdam on CIFAR-100 by 7.57, 8.67, and 0.81 percentage points, respectively. Under low-noise settings, AO-FRL requires communication comparable to or even lower than FedAvg. Privacy is further evaluated via decoder-based reconstruction PSNR, indicating reasonable protection at the selected operating point.

II. Related Work

A. Federated Learning

As a distributed machine learning paradigm, FL enables collaborative model training through interactions between clients and a central server while ensuring that raw data never leave local devices. One of the earliest and most representative approaches is FedAvg [1], which aggregates model parameters updated by clients during local training at the server side. However, this method suffers from significant performance degradation when client data follow a non-IID distribution [2]. To address this issue, a series of methods such as FedProx [4] and SCAFFOLD [5] have been proposed. Nevertheless, these approaches fundamentally still rely on local training at the client side and subsequent parameter aggregation. FedSyn introduces a data synthesis strategy based on Generative Adversarial Networks (GANs) to augment training data, and further enhances the generation process by transmitting GAN parameters with Laplacian noise [6]. FedBis, on the other hand, trains a binary classifier to infer user preferences on the client side, and aggregates the classifier parameters to provide preference-aware guidance for server-side training [7]. However, these methods typically impose considerable computational burdens on clients, especially when large-scale models are involved.

In contrast, our work adopts a representation-sharing paradigm, where clients only extract lightweight embeddings using an encoder and upload privacy-gated representations. This design significantly reduces local computation and effectively mitigates model aggregation instability in non-IID scenarios.

B. Privacy-preserving Techniques

In terms of privacy protection, another widely adopted technique is Differential Privacy (DP) [8], which is designed to mitigate the risk that sensitive information may still be inferred through auxiliary knowledge even after data anonymization. A representative extension of this idea is Differentially Private Stochastic Gradient Descent (DP-SGD), which incorporates carefully calibrated noise into gradient updates to enable privacy-preserving deep learning.

In addition to DP-based methods, another line of research focuses on preventing sensitive information leakage through data synthesis and augmentation. These approaches leverage generative models such as GANs, Variational Autoencoders (VAEs), and diffusion models to produce synthetic data that approximate the distribution of raw data. However, such methods typically require a large amount of original data to avoid overfitting during the generation process, which imposes stringent requirements on client-side data availability [9]. Moreover, they often demand substantial computational resources on the client side, making them impractical for resource-constrained environments.

POPri adopts Direct Preference Optimization (DPO) [10] to fine-tune large language models (LLMs) based on feedback collected from clients, and generates differentially private synthetic data to mimic private client data. This approach has been shown to maintain stable performance even under varying client participation rates. PrE-Text, on the other hand, constructs nearest-neighbor histograms computed locally by each client and leverages a server-side LLM to generate large-scale synthetic data for training. By doing so, it significantly reduces both communication overhead and client-side computational costs while preserving privacy [11].

In contrast to the aforementioned methods, our approach enforces privacy protection at the representation level. AO-FRL allows clients to share only differentially private embeddings generated by injecting Gaussian noise into local representations, thereby avoiding the need for local model training or generative modeling while substantially reducing privacy risks.

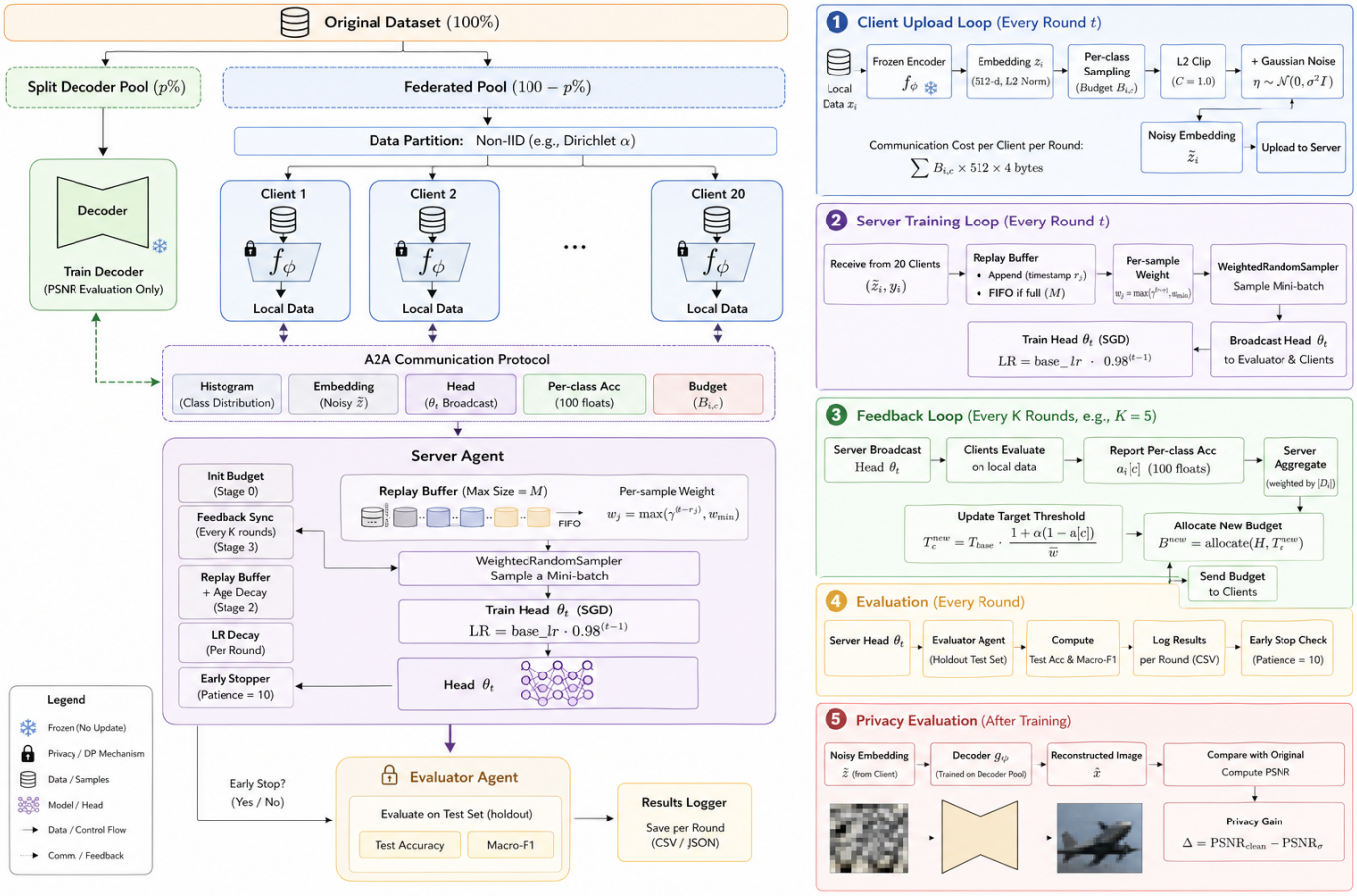

Fig. 1. Overall workflow of AO-FRL. Clients upload DP-protected embeddings to the server, which trains a global head with replay-buffer sampling. Periodic feedback adapts class-wise budgets, while utility and privacy are evaluated by test performance and reconstruction PSNR.

C. AI Agents

With the rapid development of AI agents [12], it has become possible to automate complex tasks while enhancing system flexibility through interactions among multiple agents. For instance, agent skills [13] enable modularization of agent capabilities and help reduce token consumption during task execution. Hooks provide mechanisms to inject customized logic into different stages of the agent lifecycle, thereby improving deterministic control and reliability when deploying agents. Earlier work such as the Model Context Protocol (MCP) [14] established a standardized framework for agents to invoke external tools and resources, allowing agents to access external systems in a stable and secure manner. In addition, Google introduced the Agent-to-Agent (A2A) protocol [15], which provides a standardized communication layer for heterogeneous agents, enabling platform-agnostic collaboration and reducing integration overhead.

Our work introduces an agent-oriented learning framework inspired by federated learning, in which both clients and the server are modeled as autonomous agents. Client agents are responsible for local representation extraction and reporting, while the server agent dynamically allocates upload budgets based on global distribution gaps and performance feedback. This design enables flexible, interpretable, and automated orchestration of representation sharing, fundamentally differing from conventional static federated-learning-based approaches.

III. Method

AO-FRL is a multi-agent image classification framework inspired by federated learning. AO-FRL consists of client agents, a server agent, and an evaluator agent. Client agents represent local data owners, the server agent coordinates privacy-preserving representation uploads and trains the classifier, and the evaluator agent assesses model performance.

Agents communicate through an A2A protocol [15], which enables structured and auditable client–server exchanges. Unlike conventional FL, AO-FRL avoids iterative client-side training and parameter aggregation. Instead, clients upload privacy-preserving feature representations under server-guided budgets. This design not only reduces client-side computation but also improves robustness under non-IID data distributions.

Given K client agents denoted by $\{C_1, C_2, \ldots, C_K\}$, the local dataset of client C_k is defined as

$$\mathcal{D}_k = \left\{ \left(x_i^k, y_i^k \right) \right\}_{i=1}^{n_k},$$

where x_i^k is an image, $y_i^k \in \{1, 2, \ldots, M\}$ denotes its class label, M is the number of classes, and n_k is the number of local samples on client C_k. Due to the non-IID setting, both n_k and the empirical class distribution of $\mathcal{D}_k$ vary across clients.

A. Client Statistics and Privacy-Preserving Embedding

At the start of the training, each client agent sends its class histogram (the number of images in each class) to the server agent. This histogram only provides coarse distributional information and is used by the server to allocate class-wise upload budgets. Raw images remain stored locally throughout the training process.

Given the server-assigned budget, each client performs *per-class sampling without replacement*, ensuring that the embedding of each selected image is uploaded at most once within a communication round. For each sampled image $x_j \in \mathcal{D}_i$ held by client i, the client first converts it into a feature embedding using an image encoder $g_\phi(\cdot)$:

$$\mathbf{z}_j = g_\phi(x_j). \tag{1}$$

Since the server-side classifier is trained only on uploaded embeddings rather than raw images, the structure of the embedding space may affect the effectiveness of server-side training. Ideally, samples from the same class should be located closer to each other in the embedding space, while samples from different classes should be more distinguishable.

To this end, we examine whether the adaptation of image encoders with task-related auxiliary data can produce more class-discriminative embeddings and thereby influence server-side training. We consider three encoder configurations. The default encoder is based on PyTorch's ResNet-18 model [16] pretrained on ImageNet-1K.

1) The first configuration uses the default encoder without further adaptation.
2) The second fine-tunes the encoder with a supervised cross-entropy objective, encouraging the representation to preserve information relevant to class prediction.
3) The third fine-tunes the encoder with a supervised contrastive objective, which explicitly encourages samples from the same class to be closer and samples from different classes to be farther apart.

These variants allow us to examine how different encoder adaptation strategies affect the quality of uploaded embeddings and the subsequent server-side classifier training.

For the supervised cross-entropy variant, a temporary classification head is appended to the encoder and optimized on an auxiliary dataset:

$$\mathcal{L}_{\mathrm{CE}} = -\sum_j \log p(y_j \mid x_j) + \lambda \|\theta - \theta_{\mathrm{pre}}\|_2^2. \tag{2}$$

The second term anchors the encoder parameters toward the pretrained initialization θ_{pre}, reducing excessive drift when the auxiliary dataset is limited.

For the supervised contrastive variant, the encoder is optimized to improve the geometry of the embedding space:

$$\mathcal{L}_{\mathrm{SupCon}} = -\sum_i \frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_p/\tau)}{\sum_{a \in A(i)} \exp(\mathbf{z}_i \cdot \mathbf{z}_a/\tau)}. \tag{3}$$

Here, $P(i)$ denotes the set of same-class positive samples for anchor i within a batch, $A(i)$ denotes the comparison set, and τ is the temperature. This objective function explicitly promotes intra-class compactness and inter-class separation in the embedding space. In both fine-tuning schemes, the auxiliary head used during adaptation is removed before federated training, and the resulting encoder is deployed on client agents for embedding extraction.

Before transmission, the embedding is clipped by its ℓ_2 norm:

$$\bar{\mathbf{z}}_j = \mathrm{clip}_C(\mathbf{z}_j) = \frac{\mathbf{z}_j}{\max\left(1, \frac{\|\mathbf{z}_j\|_2}{C}\right)}, \tag{4}$$

where C is the clipping threshold. This operation guarantees $\|\bar{\mathbf{z}}_j\|_2 \le C$. The clipping step normalizes the representation scale across clients, prevents embeddings with unusually large norms from dominating training, and provides a bounded representation norm for principled noise calibration.

We instantiate the Gaussian mechanism at the client side to provide per-record differential privacy. The released representation for sample x_j is defined as

$$\mathcal{M}(x_j) = \bar{\mathbf{z}}_j + \boldsymbol{\eta}, \qquad \boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d), \tag{5}$$

where d is the embedding dimension and σ controls the noise scale. Since the clipped embedding has bounded ℓ_2 norm, the sensitivity of the release is controlled by the clipping threshold C. According to the standard Gaussian mechanism, the noise scale can be calibrated as

$$\sigma \ge \frac{C\sqrt{2\ln(1.25/\delta)}}{\varepsilon}, \tag{6}$$

which ensures that each individual release satisfies (ε, δ)-differential privacy [3].

Differential privacy reduces the risk that the server can infer or reconstruct the original image from the uploaded representation. It also makes the perturbation process controllable and verifiable: the privacy-utility trade-off is explicitly governed by the clipping threshold C, the noise scale σ, and the privacy parameters ε and δ, rather than relying on heuristic noise injection.

Finally, each client uploads only the noisy embeddings and their corresponding labels to the server. The server trains the classifier using these privacy-preserving representations, while both raw images and clean embeddings remain private on the client side.

B. Initial Budget Allocation

At the beginning of training, each client C_i uploads only its local label histogram $\mathbf{h}_i \in \mathbb{N}^M$, where M is the number of classes. The server stacks all client histograms into a histogram matrix $\mathbf{H} \in \mathbb{N}^{K \times M}$, where K is the number of clients and

TABLE I
SUMMARY REPORTED BY A CLIENT TO THE SERVER.

Label Histogram (round 0 only)	**Per-class Local Validation Summary (every few rounds)**
Example values (Client 0):	**Example values (Client 0):**
Class 2: 19 samples	Class 32: 0.85 accuracy, 198 samples
Class 32: 198 samples	Class 78: 0.45 accuracy, 18 samples
Class 99: 166 samples	Class 99: 0.30 accuracy, 166 samples

$H[i, c]$ denotes the number of samples of class c owned by client C_i.

The server then computes a budget matrix $\mathbf{B} \in \mathbb{N}^{K \times M}$, where $B[i, c]$ is the maximum number of embeddings of class c that client C_i should upload in the current round. For each class c, the server first identifies the set of clients that hold samples of this class and assigns the initial budget as

$$B[i,c] = \begin{cases} \min\left(\left\lfloor \frac{T_c}{|\mathcal{S}_c|} \right\rfloor, H[i,c]\right), & i \in \mathcal{S}_c, \\ 0, & i \notin \mathcal{S}_c, \end{cases} \quad (7)$$

$$\mathcal{S}_c = \{i \mid H[i,c] > 0\}.$$

where T_c is the target number of uploaded embeddings for class c, which is initialized as $T_c = T_{\text{base}}$ (the default per-class upload target) and $\mathcal{S}_c$ denotes the set of clients that contain samples of class c. The term $T_c/|\mathcal{S}_c|$ represents the average upload quota assigned to each holder of class c, while the $\min(\cdot)$ operation ensures that no client is asked to upload more samples than it owns.

After this equal allocation, some budget may remain unassigned because certain clients have fewer samples than their allocated quota. The server redistributes this remaining budget to other clients in $\mathcal{S}_c$ that still have unused samples of class c, until the target T_c is reached or all available samples of this class are exhausted.

This histogram-based allocation mitigates the impact of non-IID client distributions. Instead of assigning the same upload quota to every client, the server allocates class-wise budgets according to each client's actual class availability. As a result, the aggregated noisy embeddings at the server are encouraged to be more class-balanced, while the upload request always remains feasible for each client.

C. Feedback-driven Budget Updating

The server periodically updates the upload budget after a fixed number of communication rounds, in order to keep the uploaded data aligned with the current weakness of the classifier. During each budget update, the server broadcasts the current classifier head parameters θ to all clients. Each client evaluates the head on its local validation embeddings and computes a per-class validation summary. As shown in Table I, the client only reports lightweight statistics, including the per-class validation accuracy and the corresponding number of validation samples. The local raw data and embeddings remain private.

The server aggregates these statistics into a global per-class validation accuracy a_c. Specifically, for each class c, the server computes a weighted average of the client-reported accuracies, where each client is weighted by its number of validation samples in that class. Therefore, clients with more validation samples contribute more reliably to the global estimate.

Based on the aggregated accuracy, the server updates the per-class upload target as:

$$T_c^{\text{new}} = T_{\text{base}} \cdot \frac{1 + \alpha(1 - a_c)}{\frac{1}{M}\sum_{j=1}^{M}[1 + \alpha(1 - a_j)]}. \quad (8)$$

where M is the number of classes, T_{base} is the default per-class upload target, a_c is the aggregated validation accuracy of class c, and α controls the feedback strength. A lower a_c gives class c a larger target in the next round, so poorly learned classes are assigned more upload budget. The denominator normalizes the targets so that the average upload target remains close to T_{base}, keeping the overall communication budget stable.

After updating the target vector, the server reruns the histogram-based budget allocation using the histogram matrix $\mathbf{H}$. This produces the next-round budget matrix, enabling the server to request more embeddings from weak classes while still respecting each client's local data availability.

D. Server-side Head Training with Replay Buffer

The server maintains a replay buffer to retain noisy embeddings uploaded in previous rounds. Each stored sample is associated with its upload round r_j.

To balance recent and historical data, the server assigns an age-based sampling weight to each sample in the buffer:

$$w_j^{(t)} = \max\left(\gamma^{t - r_j}, w_{\min}\right). \quad (9)$$

where $w_j^{(t)}$ is the sampling weight of sample j at round t, γ is the decay factor, and $w_{\min}$ is the minimum sampling weight. In our implementation, $\gamma = 0.995$ and $w_{\min} = 0.3$.

Recent samples have larger weights and are sampled more frequently, so the classifier can quickly adapt to the latest budget allocation and focus more on weak classes. Historical samples are gradually down-weighted while they remain in the buffer, which preserves useful accumulated information. The server then uses a weighted random sampler to train the classifier head for several epochs with standard supervised learning.

E. Privacy Evaluation with Decoder-based Reconstruction

To evaluate privacy protection, we assume that the server attempts to reconstruct the original image from a client-uploaded noisy embedding. The server trains a decoder using an auxiliary holdout dataset that is visually similar to the client data but has no overlap with any training data used in AO-FRL. The decoder learns to map an embedding back to an RGB image.

After training, the server feeds a noisy embedding into the decoder and obtains a reconstructed image $\hat{x}$. We use PSNR (Peak Signal-to-Noise Ratio) [17] to measure the visual similarity between the original image x and the reconstruction $\hat{x}$. PSNR reflects the relative strength between the original image signal and the reconstruction error. A higher PSNR indicates that $\hat{x}$ is closer to x, meaning more visual information is leaked; a lower PSNR indicates stronger privacy protection.

IV. Experiment

A. Experiment Setup

a) Dataset and Non-IID Data Partition: All experiments are conducted on the CIFAR-100 dataset, which contains 60,000 color images from 100 object categories, including 50,000 training images and 10,000 test images. The original training set is further divided into two parts: a *federated pool* with 45,000 images and an *auxiliary public dataset* with 5,000 images. The auxiliary public dataset is constructed by uniformly selecting 10% of the training samples from each class, so that all 100 classes are evenly represented.

To simulate realistic federated learning scenarios with heterogeneous client data distributions, we partition the federated pool among 20 client agents using a Dirichlet-based label skew strategy with concentration parameter $\alpha = 0.3$. This setting introduces random and non-IID class distributions across clients, creating a moderately challenging federated learning environment. The global test set of 10,000 images is kept separate and is never used during training.

b) Model Architecture: The machine learning models used in AO-FRL include an encoder, a decoder, and a server-side classification head. Since the original CIFAR-100 images have a resolution of $3 \times 32 \times 32$, each image is resized to $3 \times 224 \times 224$ before being fed into the encoder. We use a ResNet-18 backbone pretrained on ImageNet-1K as the shared encoder, removing its final classification layer to obtain 512-dimensional features. This resizing operation allows the encoder to extract richer visual representations from the input images.

The learned representations are reconstructed into images using a decoder that takes a 512-dimensional embedding as input. Its architecture consists of one fully connected layer followed by three transposed convolutional layers. Each embedding is mapped back to an image-space representation with dimensions $3 \times 32 \times 32$, matching the original CIFAR-100 image format. The reconstruction objective is optimized using the mean squared error (MSE) loss.

On the server-agent side, we employ a MLP classification head as the classification model. It takes the 512-dimensional embeddings uploaded by client agents as input and projects them through a hidden layer with 256 units to produce a 100-dimensional prediction over the CIFAR-100 classes. The classification objective is optimized using the cross-entropy loss. All experiments are conducted on a unified hardware environment equipped with an NVIDIA L4 GPU (24GB VRAM, 121 TFLOPS FP16), which is used for both client-side inference and server-side training.

B. Experimental Results

a) AO-FRL vs Federated Learning Baselines: Figure 2 compare AO-FRL with federated baselines trained with 10 local epochs per communication round on CIFAR-100 under non-IID Dirichlet partitioning ($\alpha = 0.3$) with 20 clients and up to 80 communication rounds. The gradient-channel baselines operate on clean local embeddings, while AO-FRL is evaluated with $\sigma \in 0, 0.005, 0.02, 0.05$, where $\sigma = 0$ removes DP perturbation. As shown in Figure 2, AO-FRL converges much faster than the baselines. The noise-free setting reaches 66.08% accuracy at round 2, while $\sigma = 0.02$ achieves the best AO-FRL accuracy of 66.66% at round 8, only 0.59 percentage points below the centralized upper bound of 67.25%. In contrast, FedAdam, the strongest federated baseline, reaches 65.85% only at round 80, while FedAvg and FedProx achieve 59.09% and 57.99%, respectively.

Table II shows the same trend for macro-F1. AO-FRL at $\sigma = 0.02$ reaches 66.44% at round 8, outperforming FedAvg 58.58% and FedProx 57.39%. FedAdam attains 65.66% at round 80, slightly below the best AO-FRL setting. These results indicate that AO-FRL improves not only overall accuracy but also class-balanced performance under heterogeneous data distributions. Table II further shows that AO-FRL achieves communication and time efficiency through faster convergence. Benefiting from the replay-buffer mechanism, AO-FRL aggregates and reuses uploaded embeddings at the server, allowing training to converge in far fewer communication rounds despite its larger per-round payload; for example, $\sigma = 0.02$ achieves 66.66% accuracy with only 1,694 MB of communication, compared with 2,017 MB for FedAdam. This also translates into shorter wall-clock time at low-to-moderate noise: AO-FRL stops in 2m33s, 2m57s, and 4m49s for $\sigma = 0$, $\sigma = 0.005$, and $\sigma = 0.02$, respectively, whereas FedAvg, FedProx, and FedAdam run for 14m56s, 23m30s, and 15m05s without early stopping. Under stronger noise, however, $\sigma = 0.05$ requires 50 rounds, increasing communication to 4,684 MB and training time to 16m56s, reflecting the privacy–communication–time trade-off.

b) PSNR-Based Privacy Evaluation: To quantitatively assess the privacy protection introduced by DP Gaussian noise under controlled conditions, we use PSNR to measure reconstruction quality.

Table III reports the median reconstruction PSNR on the CIFAR-100 test set under three DP noise levels, together with the noise-free baseline, using the ImageNet-frozen encoder.

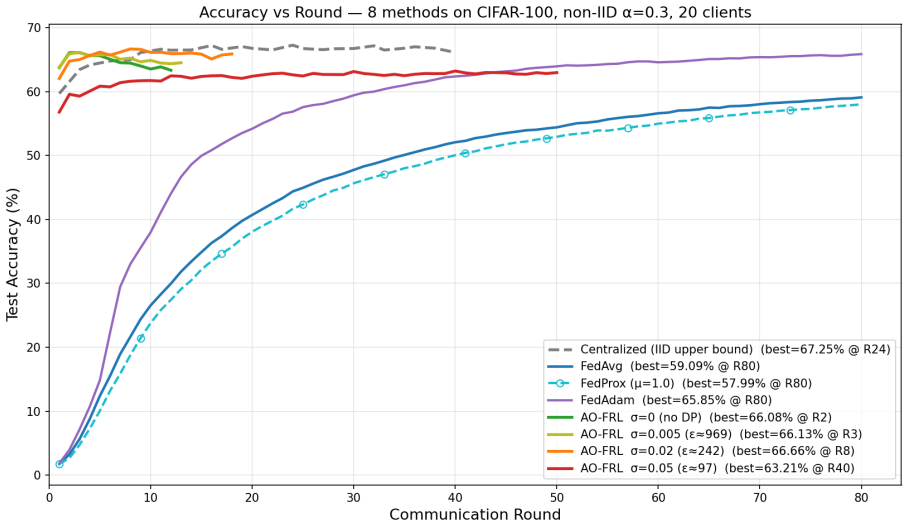

Fig. 2. Accuracy on CIFAR-100 under non-IID Dirichlet $\alpha = 0.3$. AO-FRL reaches near-centralized accuracy 66.1–66.7% within 2–8 rounds, outperforming FedAvg 59.1% and FedProx 58.0%. Showing AO-FRL's faster convergence and stronger overall performance.

TABLE II
PERFORMANCE SUMMARY COMPARISON

Method	Best Acc	Best F1	Comm. at Conv.
Centralized	**67.25%** (24)	**67.33%** (16)	0 MB
AO-FRL σ=0	66.08% (2)	66.14% (3)	**1132 MB**
AO-FRL σ=0.005	66.13% (3)	66.127% (3)	1223 MB
AO-FRL σ=0.02	**66.66%** (8)	**66.44%** (8)	1694 MB
AO-FRL σ=0.05	63.21% (40)	62.88% (40)	4684 MB
FedAdam	65.85% (80)	65.66% (80)	2017 MB
FedAvg	59.09% (80)	58.58% (80)	2017 MB
FedProx (μ=1.0)	57.99% (80)	57.39% (80)	2017 MB

TABLE III
RECONSTRUCTION PSNR USING AN IMAGENET-PRETRAINED ENCODER, ACROSS 3 DP NOISE LEVELS AND A CLEAN BASELINE. LOWER MEDIAN PSNR INDICATES STRONGER PIXEL-LEVEL PRIVACY.

Encoder	σ	ε	Median PSNR (dB)
ImageNet pretrained	0	∞	12.25
	0.005	1060	12.21
	0.020	265	11.80
	0.050	106	10.75

The noise-free setting achieves a median PSNR of 12.25 dB, indicating that the 512-dimensional embedding bottleneck already imposes substantial information loss. As the Gaussian noise scale increases, the median PSNR decreases monotonically: $\sigma = 0.005$ yields only a marginal reduction of 0.03 dB, while $\sigma = 0.02$ and $\sigma = 0.05$ reduce PSNR by 0.43 dB and 1.53 dB, respectively. This suggests that stronger noise improves reconstruction resistance, although the effect grows sub-linearly because much of the pixel-level information has already been discarded by the encoder.

c) Dynamic Budget Allocation: To analyze how the server agent guides budget reallocation across client agents, we track the standard deviation of the per-class upload budget

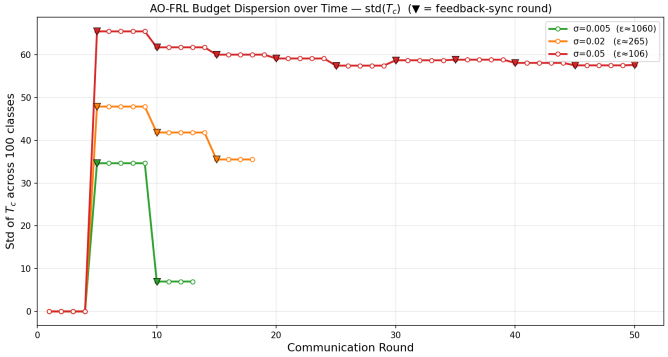

Fig. 3. Per-class budget dispersion (standard deviation (std) of T_c) across rounds — collapses at low σ as the federation converges

T_c over the 100 CIFAR-100 classes. This metric reflects the dispersion of class-wise upload budgets and indicates how actively the feedback mechanism redistributes upload capacity. As shown in Figure 3, all AO-FRL variants start from the uniform budget $T_{\text{base}} = 500$ and remain unchanged before the first synchronization. Since client agents communicate with the server agent every 5 rounds to maintain training quality, the first budget update occurs after round 5, where T_c is adjusted based on per-class validation accuracy and the budget dispersion increases accordingly.

The observed trajectories show that DP noise affects the budget allocation process. Under $\sigma = 0.005$, the dispersion quickly collapses as the classifier converges and per-class accuracies become more balanced. In this case, the feedback signal weakens because there are no longer clear underperforming classes that require additional uploads. Under $\sigma = 0.02$, the dispersion decreases more slowly and remains moderate, indicating that some class-level performance gaps persist. In contrast, under $\sigma = 0.05$, the dispersion does not

TABLE IV
RECONSTRUCTION PSNR ON THE CIFAR-100 TEST SET UNDER DIFFERENT ENCODER SETTINGS. THE IMAGENET ENCODER IS USED WITHOUT FINE-TUNING UNLESS OTHERWISE SPECIFIED.

Encoder setting	σ	PSNR median (dB)
ImageNet pretrained	–	12.25
ImageNet pretrained + Gaussian noise	0.05	10.75
ImageNet fine-tuned (CE + L2 anchor)	0.05	10.87
ImageNet fine-tuned (SupCon)	0.05	11.89

collapse. The stronger Gaussian noise lowers the signal quality of uploaded embeddings, leaving persistent per-class accuracy differences; as a result, the server agent continues to assign larger budgets to weaker classes throughout training.

C. Ablation Study

a) Encoder Ablation: To assess whether a fine-tuned encoder improves the final training performance, we conduct experiments using 5K auxiliary samples. Figure 4 reports AO-FRL (σ=0.02) with three encoders: ImageNet-frozen, Scheme A (CE + L2 anchor), and Scheme B (SupCon). Compared to CE-based fine-tuning, SupCon achieves substantially higher accuracy and faster convergence. ImageNet-frozen and SupCon achieve similar peaks (66.66% vs. 66.20%, within run variance) with consistent macro-F1 ranking (66.44% vs. 65.88%). Scheme A underperforms (61.01%, -5.7%) and converges slower (peak at round 11 vs. 8 for frozen). SupCon converges fastest, reaching its peak at round 5 and early-stopping at round 15, reducing communication to 1.42 GB (-16% vs. frozen).

Additionally, according to Table IV, privacy also varies across encoders under the same noise level (σ=0.05): SupCon yields the highest PSNR (11.89 dB), followed by CE (10.87 dB) and the frozen encoder (10.75 dB).

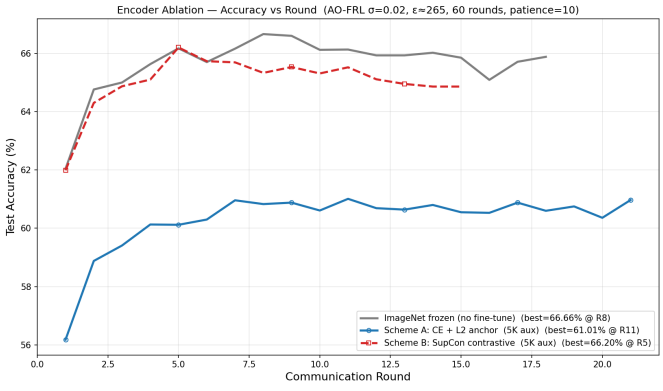

Fig. 4. Test accuracy vs. communication rounds for AO-FRL (σ=0.02). ImageNet-frozen and SupCon achieve similar peaks around 66.5%, whereas CE+anchor fine-tuning degrades by 5.7%; SupCon converges fastest at R5.

b) Agent-Orchestrated Ablation: To analyze the role of the server agent in dynamic budget allocation, we conduct an ablation study with the replay buffer disabled, so that the server is trained only on newly uploaded data in each round.

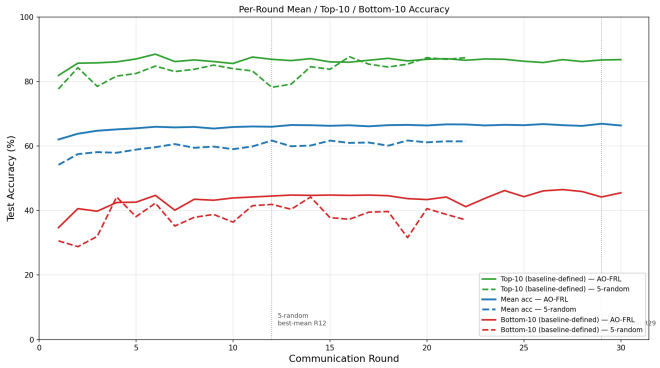

Fig. 5. Per-round mean (blue line), top-10 (green line), and bottom-10 (red line) test accuracy of $K = 5$ partial participation without server-side budget management versus full AO-FRL ($T_{\text{base}} = 500$, replay enabled).

We compare two configurations matched in total upload volume: full AO-FRL with all 20 clients and server-coordinated per-class budget allocation, and a $K = 5$ partial-participation baseline where randomly selected clients upload an equivalent number of samples without server-side class allocation.

By Figure 5, full AO-FRL achieves 66.66% mean accuracy, compared with 61.18% for the un-orchestrated baseline, yielding a 5.48% gain. The improvement is concentrated in the long tail: the most improved accuracy increases from 33.6% to 55.2% (+21.6%), the bottom-10 mean rises from 37.56% to 45.64% (+8.08%), and the per-class standard deviation decreases from 15.13 to 13.49. While the top-10 easiest classes remain at a high accuracy level of 86.48%. This suggests that server-side budget management primarily improves difficult classes rather than uniformly shifting all classes upward. The bottom-10 classes are dominated by visually confusable categories, such as flatfish, shrew, bowl, and baby, which require more disambiguating samples. By contrast, the top-10 classes are visually distinctive objects, such as motorcycle (93%), wardrobe (90%), and sunflower (88%).

The convergence trajectories further highlight the benefit of server orchestration. Full AO-FRL reaches 62.04% accuracy in a single round and 65.50% by round 5, whereas the un-orchestrated baseline requires 12 rounds to reach 61.71% and then plateaus, triggering early stopping at round 22. The same pattern is more pronounced for the bottom-10 classes: AO-FRL reaches nearly 30% long-tail accuracy in round 1, while the baseline fluctuates between 10% and 25% throughout its 22-round run. Thus, dynamic server-side budget management provides three benefits: higher final accuracy, monotonic and stable convergence without the early-peak collapse seen under random partial participation, and stronger protection against long-tail collapse for visually confusable classes

D. Evaluation on Alternative Datasets

To evaluate whether AO-FRL generalizes beyond CIFAR-100, we apply the same pipeline to CIFAR-10 and SVHN [18], using a frozen ImageNet encoder, noise-free image embeddings, and the replay buffer, under Dirichlet non-IID partition-

TABLE V
RESULTS AND DATASET CHARACTERISTICS ACROSS ALTERNATIVE DATASETS (NO DP, WITH REPLAY BUFFER, AND DIRICHLET $\alpha = 0.3$).

Dataset	Classes	Train Size	AO-FRL Acc (R)	Centralized Acc (R)
CIFAR-100	100	50K	66.08% (2)	67.25% (24)
CIFAR-10	10	50K	87.47% (27)	88.04% (46)
SVHN	10	73K	67.77% (16)	72.48% (34)

ing with $\alpha = 0.3$. As shown in Table V, AO-FRL converges faster than centralized training on all three datasets—CIFAR-10, CIFAR-100, and SVHN—benefiting from the replay buffer. Centralized training achieves accuracies of 88.04%, 67.25%, and 72.48%, while AO-FRL reaches 87.47%, 66.45%, and 67.77%, respectively.

CIFAR-10 achieves the higher accuracy, benefiting from its 10-class natural-image structure and strong alignment with the ImageNet-pretrained encoder. SVHN converges stably despite its digit-domain images being less aligned with ImageNet features. These results show that AO-FRL is not tied to CIFAR-100 and remains effective across different label spaces and visual domains under the same training pipeline.

V. DISCUSSION

A. Replay Buffer as Non-Parametric Server Memory

AO-FRL differs from FedAvg in how information is retained across rounds. While FedAvg compresses past client contributions into model parameters and discards the underlying data, AO-FRL maintains a server-side replay buffer of embeddings, enabling reuse of historical information during training. This design introduces a fundamental trade-off. The accumulated embeddings effectively enlarge the training set, enabling much faster convergence than approaches without such memory. However, repeatedly reusing and aggregating embeddings may allow the server to partially suppress the injected noise and recover more stable representations, increasing the risk of information leakage. Thus, the replay buffer improves efficiency and convergence at the potential cost of privacy.

B. Effect of Non-IID Data on AO-FRL

The robustness of AO-FRL to label-distribution heterogeneity is evaluated on CIFAR-100 by varying the Dirichlet concentration parameter $\alpha \in \{0.1, 0.3, 100\}$. All runs use 20 clients, DP noise with $\sigma = 0.02$, and early stopping with patience 10. AO-FRL's peak accuracy appears to be only weakly affected by α. The best test accuracies are highly consistent across substantially different non-IID levels: 66.56% for $\alpha = 0.1$, 66.66% for $\alpha = 0.3$, and 66.02% for $\alpha = 100$. The total variation is only 0.64 percentage points, suggesting that final performance is largely decoupled from the degree of label heterogeneity. This is notable because $\alpha = 0.1$ corresponds to a highly non-IID long-tail setting, whereas $\alpha = 100$ is close to IID.

This robustness is likely due to AO-FRL's upload-based design. Since class-wise upload budgets are allocated globally and redistributed among the clients holding each class, the effective number of server-visible samples per class remains relatively stable despite changes in client-level label distributions. Therefore, AO-FRL's performance ceiling appears to be determined more by the encoder representation and the server-side head capacity than by the specific data partition.

A secondary observation is that convergence becomes slightly slower as α increases: the peak is reached at round 6 for $\alpha = 0.1$, round 8 for $\alpha = 0.3$, and round 11 for $\alpha = 100$. A conservative explanation is that stronger label imbalance may make sampling easier, as each class is concentrated on fewer clients and its budget can be filled more quickly. This may explain the earlier convergence under smaller α.

C. Encoder Fine-Tuning with Limited Data

As shown in Figure 4, the encoder ablation highlights an important transfer-learning principle: fine-tuning a strong pretrained encoder on limited auxiliary data may degrade, rather than improve, downstream performance. With only 5K auxiliary samples, supervised fine-tuning provides insufficient coverage of the full data distribution and can cause the encoder to over-specialize to the auxiliary subset.

This effect is evident for cross-entropy-based fine-tuning, which directly adapts the encoder toward class-specific decision boundaries and may overwrite broadly useful ImageNet features. In contrast, supervised contrastive fine-tuning preserves the relative geometry of the embedding space more effectively, leading to better performance than cross-entropy fine-tuning. However, it still does not surpass the frozen ImageNet encoder. Thus, when the fine-tuning data are limited, encoder adaptation may underperform the original pretrained model. Since the server agent has no access to the clients' raw data in our setting, fine-tuning with a small amount of similar auxiliary data does not improve the final training performance.

D. Limitations and Future Work

Our PSNR analysis suggests that the embedding transformation may contribute more to privacy protection than the added Gaussian DP noise. Mapping high-dimensional images into compact frozen-encoder embeddings substantially limits reconstructable visual information, especially when the decoder is trained with limited auxiliary data. Nevertheless, increasing the noise level still consistently reduces PSNR, confirming that DP noise provides additional privacy protection. This comes with a utility cost: stronger noise requires more training rounds and communication to reach comparable performance.

A future direction is to make this utility-privacy trade-off explicit through architectural design. For example, a stricter streaming variant could discard uploaded embeddings after each round, preventing cross-round aggregation but sacrificing the convergence benefits of replay. More generally, bounded replay windows may offer intermediate operating points between training efficiency and privacy protection.

Extending AO-FRL to more complex and sensitive domains would further demonstrate its practical value. Promising directions include multi-modal medical analysis, federated

video understanding, and financial fraud detection. In such scenarios, the multi-agent orchestration of AO-FRL enables fully automated training and decision-making, reducing human involvement in data handling and policy configuration. This automation not only improves scalability but also strengthens privacy protection by minimizing the risk of human-induced information leakage and ensuring consistent enforcement of privacy-preserving protocols.

VI. Conclusion

This paper proposes AO-FRL, a learning framework inspired by federated learning but fundamentally built upon multi-agent communication and dynamic orchestration rather than traditional federated learning paradigms. AO-FRL enables automated model training by allowing agents to collaboratively exchange representations while adapting to local data characteristics. Through server-side orchestration, the framework dynamically adjusts the upload budget of each client according to the degree of non-IID data heterogeneity, ensuring sufficient data diversity and balanced class representation.

Experiments show that this orchestration mechanism leads to clear empirical gains. On CIFAR-100 with 20 non-IID clients, AO-FRL ($\sigma = 0.02$) has the best test accuracy of 66.66%, outperforming FedAvg (59.09%), FedProx (57.99%), and FedAdam (65.85%) by 7.6, 8.7, and 0.8 percentage points, respectively. AO-FRL also converges substantially faster, surpassing all three federated baselines within 40 rounds and reaching its peak at round 8. In wall-clock time, AO-FRL converges within 2m33s–4m49s under low-to-moderate noise levels, whereas FedAvg, FedProx, and FedAdam require 14m56s–23m30s without early stopping. In terms of communication, AO-FRL remains comparable to or lower than Fed-style baselines in the same regime, benefiting from its much faster convergence.

Beyond the overall performance gains, the ablation study further demonstrates the effectiveness of dynamic per-class budget allocation. This mechanism improves long-tail learning under non-IID partitions by increasing coverage for under-represented and difficult classes. Specifically, bottom-10 accuracy increases by 8.08%, and the accuracy of the most improved class rises from 33.6% to 55.2%, while the top-10 classes remain nearly unchanged (+0.16%). Moreover, AO-FRL generalizes beyond CIFAR-100, where for the CIFAR-10 and SVHN datasets, AO-FRL has accuracies close to those of the centralized training with faster convergence.

Overall, AO-FRL demonstrates a practical and promising direction for building automated, privacy-preserving, and adaptive learning systems through the integration of multi-agent collaboration and federated learning principles.

The source code of this paper is available at: https://github.com/uwm-se/AO-FRL.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[2] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[3] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International conference on machine learning*, pages 394–403. PMLR, 2018.

[4] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of the 3rd International Conference on Machine Learning and Systems (MLSys 2020)*, 2020. FedProx was introduced as a generalization of FedAvg to address systems and statistical heterogeneity:contentReference[oaicite:1]index=1.

[5] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020.

[6] Monik Raj Behera, Sudhir Upadhyay, Suresh Shetty, Sudha Priyadarshini, Palka Patel, and Ker Farn Lee. Fedsyn: Synthetic data generation using federated learning. *arXiv preprint arXiv:2203.05931*, 2022.

[7] Feijie Wu, Xiaoze Liu, Haoyu Wang, Xingchen Wang, Lu Su, and Jing Gao. Towards federated rlhf with aggregated client preference for llms. *arXiv preprint arXiv:2407.03038*, 2024.

[8] Jordan Awan and Aishwarya Ramasethu. Optimizing noise for f-differential privacy via anti-concentration and stochastic dominance. *Journal of Machine Learning Research*, 25(351):1–32, 2024.

[9] Milad Abdollahzadeh, Touba Malekzadeh, Christopher TH Teo, Keshigeyan Chandrasegaran, Guimeng Liu, and Ngai-Man Cheung. A survey on generative modeling with limited data, few shots, and zero shot. *arXiv preprint arXiv:2307.14397*, 2023.

[10] Charlie Hou, Mei-Yu Wang, Yige Zhu, Daniel Lazar, and Giulia Fanti. Private federated learning using preference-optimized synthetic data. In *Forty-second International Conference on Machine Learning*, 2025.

[11] Charlie Hou, Akshat Shrivastava, Hongyuan Zhan, Rylan Conway, Trang Le, Adithya Sagar, Giulia Fanti, and Daniel Lazar. Pre-text: Training language models on private federated data in the age of llms. *arXiv preprint arXiv:2406.02958*, 2024.

[12] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[13] Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025.

[14] Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model context protocol (mcp): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*, 2025.

[15] Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279*, 2025.

[16] 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*, pages 770–778, 2016.

[17] Rayan Al Sobbahi and Joe Tekli. Comparing deep learning models for low-light natural scene image enhancement and their impact on object detection and classification: Overview, empirical evaluation, and challenges. *Signal Processing: Image Communication*, 109:116848, 2022.

[18] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4, 2011.