International Journal of Computer Vision (2026) 134:532
https://doi.org/10.1007/s11263-026-03125-2

FedDyMem: Efficient Federated Learning with Dynamic Memory and Memory-Reduce for Unsupervised Image Anomaly Detection

Silin Chen[1] · Andy Liu[1] · Kangjian Di[1] · Yichu Xu[2,3] · Han-Jia Ye[2,3] · Wenhan Luo[4] · Ningmu Zou[1,5]

Received: 20 December 2025 / Accepted: 30 October 2026
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2026

Abstract

Unsupervised image anomaly detection (UAD) has become a critical process in industrial and medical applications, where models are trained on normal samples without requiring labeled anomalies. However, even normal images may contain sensitive product designs, manufacturing information, or institution-specific population characteristics, making centralized data collection difficult in practical deployments. Federated UAD provides a promising decentralized solution, but it remains challenging because clients use only normal training samples, their feature distributions remain highly heterogeneous within the same task, and communication should be efficient while limiting the exposure of sensitive client information. To address these challenges, we propose FedDyMem, an efficient federated learning method with dynamic memory and memory-reduce for unsupervised image anomaly detection. Instead of exchanging full model parameters, FedDyMem facilitates knowledge sharing between the clients and server through reduced dynamic memory banks. In the local clients, a memory generator and a metric loss are employed to improve the consistency of the feature distribution for normal samples, leveraging the local model to update the memory bank dynamically. For efficient communication and data privacy, a memory-reduce method based on weighted averages is proposed to significantly decrease the scale of memory banks. This representation reduces communication overhead and empirically lowers reconstruction fidelity under inversion attacks. On the server, global memory is constructed and distributed to individual clients through k-means aggregation. Experiments on six federated UAD tasks constructed from eleven public industrial and medical datasets show that FedDyMem consistently improves anomaly detection and localization while reducing communication cost.

Keywords Unsupervised federated learning · Unsupervised image anomaly detection · Feature distribution shift · Communication efficiency

1 Introduction

Unsupervised image anomaly detection has achieved significant success in various domains, such as industrial inspection (He et al., 2024) and medical disease recognition (Cai et al., 2024). Obtaining reliable annotations for abnormal sam-

Communicated by Matej Kristan.

✉ Ningmu Zou
nzou@nju.edu.cn

Silin Chen
silin.chen@smail.nju.edu.cn

Andy Liu
andyliu@smail.nju.edu.cn

Kangjian Di
kangjiandi@smail.nju.edu.cn

Yichu Xu
xuyc@lamda.nju.edu.cn

Han-Jia Ye
yehj@lamda.nju.edu.cn

Wenhan Luo
whluo.china@gmail.com

[1] School of Integrated Circuits, Nanjing University, Suzhou 215163, Jiangsu, China

[2] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, Jiangsu, China

[3] School of Artificial Intelligence, Nanjing University, Nanjing 210023, Jiangsu, China

[4] Hong Kong University of Science and Technology, Hong Kong, China

[5] Interdisciplinary Research Center for Future Intelligent Chips (Chip-X), Nanjing University, Suzhou 215163, Jiangsu, China

Published online: 23 November 2026

ples can be time-consuming. UAD therefore builds anomaly detectors from normal training data without requiring annotations of real abnormal samples. However, existing methods typically rely on centralized training with large-scale datasets composed of normal samples. In practice, normal data in industrial and medical scenarios remain highly sensitive. For industrial inspection, normal images may contain critical intellectual property, such as PCB layouts, wafer patterns, component geometries, and material textures (Fyrbiak et al., 2017). Moreover, these images may implicitly reveal the deployed automated optical inspection (AOI) solutions, inspection logic, and acceptance criteria (Cao et al., 2025b). For medical screening, private information related to patient attributes, imaging protocols, and institutional screening populations may still be exposed even from normal images (Macpherson et al., 2023). Consequently, industrial companies and healthcare organizations often have practical limitations in collecting and aggregating raw data, which significantly challenge centralized learning approaches (Zhang et al., 2024b; Zhao et al., 2024).

Recently, federated learning has been developed as a privacy-preserving paradigm for machine learning, providing collaborative model training across distributed devices while maintaining raw data locally and not transmitting to central servers (Saha et al., 2024). Thus, the integration of local training and global aggregation strategies has garnered significant attention in industrial and healthcare domains, enabling data privacy protection (Zhao et al., 2024; Zhang et al., 2024b). We consider federated UAD among clients that share the same inspection or screening task. For example, separate federations may be formed for wafer defect detection, PCB inspection, or chest X-ray screening, rather than combining semantically unrelated anomaly detection tasks within the same federation. Each client uses only local normal samples for training, without sharing raw data. Moreover, directly averaging model parameters across heterogeneous clients may not preserve the normal representations required for anomaly scoring. Thus, federated UAD involves not only distributed optimization but also the collaborative learning of normal representations across clients. We summarize the main challenges as follows:

(1) End-to-end unsupervised anomaly detection without additional labeled data or supervised fine-tuning. Most existing unsupervised federated learning methods focus on self-supervised representation learning or prototype learning, and typically require local fine-tuning through a supervised linear evaluation protocol after federated training (Zhuang et al., 2021; Liao et al., 2024; Han et al., 2022; Zhang et al., 2024). This limits their applicability to UAD, where only normal samples are used during training.

(2) Heterogeneous normal distributions across clients. As shown in Fig. 1, although all clients address the same UAD task and use only normal samples for training, their feature distributions can vary substantially across clients due to differences in product subtypes, manufacturing conditions, imaging devices, acquisition protocols, or screening populations (Lei et al., 2025). Such heterogeneous normal modes may also arise in centralized UAD when diverse normal data are modeled jointly (You et al., 2022). The additional difficulty in the federated setting is that these normal modes are not jointly observable at the sample level during optimization. Consequently, local objectives may become specialized to incomplete normal supports. This limited local coverage may cause a model to assign high anomaly scores to normal images from other clients. Inconsistencies among local updates can hinder the construction of a federation-wide normal representation, leading to performance degradation and convergence difficulties under data heterogeneity (Li et al., 2020a; Karimireddy et al., 2020; Lei et al., 2025).

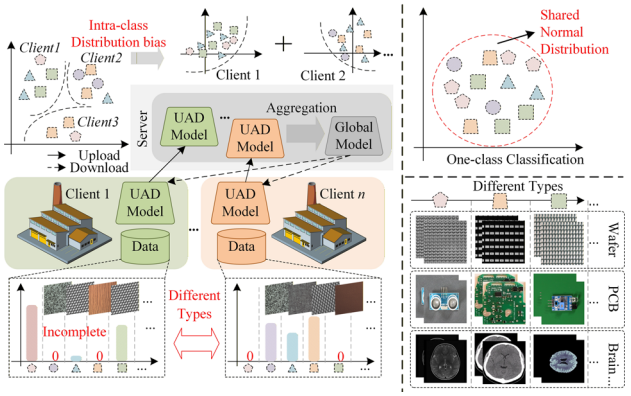

Fig. 1 Illustration of federated learning in UAD. As shown on the left, for the same inspection or screening task, an intra-class distribution bias exists between local models, caused by clients possessing different and incomplete subsets of normal data within the same inspection or screening task. As shown in the top right, federated UAD aims to construct a shared normal distribution for one-class anomaly detection across clients. Examples of type differences in anomaly detection are shown in the bottom right

3) Communication-efficient and privacy-aware knowledge sharing. Standard federated learning methods commonly exchange full model parameters or gradients, which may incur heavy communication overhead in resource-constrained industrial and medical scenarios (Li et al., 2020a; Almanifi et al., 2023). More importantly, the exchanged information may still encode product textures, imaging protocols, or patient-related distributional patterns, making it sensitive to potential reconstruction or inversion attacks (Cao et al., 2025a).

To address these issues, we introduce FedDyMem, an efficient federated learning framework with dynamic memory and memory-reduce, specifically designed for federated UAD. FedDyMem facilitates knowledge transfer in federated learning by sharing the memory bank instead of model parameters among clients. Specifically, it constructs a local memory bank using the limited dataset of normal samples

available at each client. To address the challenge of intra-class distribution bias in different clients, a memory generator and a metric loss function are introduced to improve the consistency of normal feature distributions. Then the local memory bank is dynamically updated by the local client model before uploading. To mitigate distribution bias during aggregation, FedDyMem employs k-means clustering to extract general distribution from different clients, thereby ensuring effective aggregation without confusion. Considering communication efficiency and data privacy, a memory-reduce method is proposed to decrease the scale of memory banks, thereby reducing its mutual information with the original data during dynamic updates. We then evaluate our proposed method separately on six federated UAD tasks corresponding to different product or health screening types, achieving state-of-the-art performance in each case. The main contributions are summarized as follows:

- To the best of our knowledge, FedDyMem is the first federated learning framework designed specifically for unsupervised image anomaly detection. This framework facilitates collaborative training using only normal samples for local training.
- To address the significant distribution biases of normal samples across clients, we incorporate a memory generator and a metric loss function that improve the consistency of feature distributions. Additionally, we employ k-means clustering during the aggregation phase to reduce inter-client discrepancies and mitigate potential ambiguities.
- To achieve both communication efficiency and privacy-aware knowledge sharing, we propose a memory-reduce method based on a dynamic weighted average. This method substantially decreases the size of the memory bank, thereby reducing both communication overhead and the mutual information between uploaded information and raw data, while preserving model performance.
- We construct six federated UAD tasks by grouping eleven public image anomaly detection datasets according to product or health screening types. Comprehensive experiments demonstrate that FedDyMem achieves excellent federated UAD performance.

2 Related Work

2.1 Unsupervised Image Anomaly Detection

The objective of UAD is to identify whether a given sample is anomalous and to precisely localize the anomaly regions, using a training dataset that comprises only normal samples (Xie et al., 2024). The reconstruction-based methods are significant in UAD, which assumes that anomalous samples cannot be accurately reconstructed by feature learning models trained exclusively on normal samples (Liang et al., 2023). In contrast to traditional UAD approaches, which rely on a single, centralized training dataset, federated UAD requires the efficient aggregation of information from samples distributed across multiple clients. Reconstruction-based methods, however, typically rely solely on normal samples within the training set and fail to capture key characteristics of out-of-distribution samples. When these centralized UAD objectives are directly optimized on each client in a federated setting, the learned representations may fit client-specific normal modes rather than the shared normal distribution required for federated UAD (You et al., 2022). Another widely utilized approach for anomaly detection is the memory bank-based method. SPADE (Cohen & Hoshen, 2020) proposed a semantic pyramid structure to construct a pixel-level feature memory, effectively facilitating anomaly detection. PaDiM (Defard et al., 2021) employed Gaussian distributions derived from normal samples as the memory bank and utilized the Mahalanobis distance as the anomaly metric. PatchCore (Roth et al., 2022) introduced local neighborhood aggregation to expand the receptive field of the memory bank while preserving resolution. Additionally, PatchCore implemented a greedy core-set subsampling strategy to reduce the memory bank size without significant performance degradation. PNI (Bae et al., 2023) further advanced the paradigm by integrating spatial and neighborhood information into the memory bank construction, thereby improving its capacity to represent normal samples comprehensively. These methods extract features from normal images and store them within a feature memory bank. During testing, the sample queries the memory bank to retrieve features corresponding to the k-nearest neighbors. However, the memory banks are non-trainable, limiting their capacity to learn information across local clients in federated learning. Beyond these category-specific approaches, unified UAD aims to detect anomalies from multiple semantic categories using a single model. UniAD (You et al., 2022) introduced a unified framework for multi-class anomaly detection, enabling heterogeneous categories to be jointly modeled by a single anomaly detector. MambaAD (He et al., 2024) further explored state-space models for multi-class UAD to improve the modeling of heterogeneous normal patterns. More recently, InpFormer (Luo et al., 2025) investigated intrinsic normal prototypes within individual images for universal anomaly detection, further enhancing the representation of diverse normal patterns. These studies suggest that appropriately designed unified models can mitigate inter-category interference and effectively model heterogeneous normal distributions when data from different categories are jointly available during training. However, these methods generally assume centralized access to heterogeneous normal

data, whereas each client in federated UAD observes only its local normal distribution. Therefore, learning a unified normal representation requires integrating complementary normal modes across clients through federated communication rather than direct joint training.

2.2 Federated Learning

Federated Learning with Data Heterogeneity. FedAvg (Mcmahan et al., 2017) is a foundational approach in federated learning that enables the training of a global model by aggregating parameters from locally trained models. Despite its effectiveness, the performance of FedAvg degrades significantly with high data heterogeneity. FedProx (Li et al., 2020b) mitigates client drift by introducing a proximal regularization term, while SCAFFOLD (Karimireddy et al., 2020) employs control variates to correct local update directions, thereby improving optimization under data heterogeneity. Other methods address data heterogeneity through personalized models. FedBN (Li et al., 2021a) mitigates data heterogeneity challenges in federated learning by maintaining BN parameters that are specific to each client's local model. APPLE (Luo & Wu, 2022) aggregates client models locally by learning precise weight updates instead of relying on approximations. However, these methods are mainly designed for supervised federated learning and are not directly tailored to federated UAD. In UAD, all training samples belong to the normal class, and directly combining local UAD objectives with parameter aggregation may cause each client to overfit its own normal distribution. Recent studies address this issue by sharing or synthesizing auxiliary data across clients (Nguyen et al., 2021; Li et al., 2024b). Although such strategies can improve different client distribution coverage, sharing additional data among clients may increase the risk of data privacy leakage (Li et al., 2024a). More recently, prototype-based federated learning methods have been developed to reduce generalization gaps by aligning global and local feature prototypes (Tan et al., 2022a; Zhang et al., 2024a). These methods are effective when semantic class labels are available, because category-specific prototypes can serve as explicit anchors for cross-client feature alignment. However, federated UAD only contains normal samples during training, making it difficult to construct discriminative category-specific prototypes.
Unsupervised Federated Learning. In unsupervised federated learning, most existing methods focus on self-supervised representation learning, knowledge distillation, or prototype-based representation learning, which aim to learn good feature representations from unlabeled data to facilitate downstream machine learning tasks (Zhuang et al., 2021; Liao et al., 2024; Han et al., 2022; Zhang et al., 2024). However, these methods depend on aligning local representations with global representations, a task that proves challenging for UADs constrained to a single category. ProtoFL (Kim et al., 2023) was proposed to integrate federated learning with prototyping for addressing one-class classification. However, its two-stage fine-tuning method, which utilizes normalizing flow for representation learning, introduces additional computational overhead and ignores distribution bias among clients in UAD. Furthermore, many unsupervised federated learning approaches require supervised fine-tuning, which limits their applicability to UAD.
Communication-Efficient and Privacy-Aware Federated Learning. Communication between clients in federated learning is limited by network bandwidth, unreliable connections, and varying device statuses. Consequently, designing communication-efficient algorithms is crucial to overcome these challenges (Almanifi et al., 2023). Existing methods primarily tackle this issue by either decreasing the total number of communication rounds required for model convergence or reducing the data uploaded during each communication round (Li et al., 2020a). To accelerate global convergence, some approaches leverage mechanisms such as localized model updates (Li et al., 2020b) and gradient optimization enhancements (Yang et al., 2022), effectively reducing the number of communication rounds. To reduce the data uploaded, methods such as model compression (Shlezinger et al., 2020; Li et al., 2023) and selective client participation (Xu & Wang, 2020) have been extensively explored to minimize communication overhead. Beyond efficiency, privacy-aware communication is equally essential, especially in resource-constrained and privacy-sensitive environments. The exchanged information should resist reconstruction attempts from adversaries, preventing leakage of the raw client data (Cao et al., 2025a). These studies suggest that communication reduction and privacy-aware representation sharing are both important, yet existing methods do not specifically address this for federated UAD with heterogeneous normal distributions.

3 Preliminary

3.1 Typical Federated Learning

In the typical federated learning, such as FedAvg (Mcmahan et al., 2017), there exist N clients $\{\mathcal{C}_n\}_{n=1}^{N}$ with their individual datasets $\{\mathcal{D}_n = (\mathcal{X}_n, \mathcal{Y}_n) \sim \mathbb{P}_n(\mathcal{X}, \mathcal{Y})\}_{n=1}^{N}$ and local models $\{f_n(\cdot;\theta_n)\colon \mathcal{X}_n \to \mathcal{Y}_n\}_{n=1}^{N}$, where n is the index for the client, $(\mathcal{X}_n, \mathcal{Y}_n)$ denotes a set of samples and labels on the n-th client, $\mathbb{P}_n(\mathcal{X}, \mathcal{Y})$ represent the joint distribution of samples and labels. The objective of the typical federated learning is to facilitate collaborative training of a global model θ_{glob} that generalizes effectively across the entire dataset $\mathcal{D}$ while ensuring data privacy. The global training optimization target is:

$$\min_{\theta_{\text{glob}}} \mathcal{L}(\theta_{\text{glob}}) \triangleq \sum_{n=1}^{N} z_n \mathcal{L}_n(\theta_n), \tag{1}$$

where the z_n denotes the weight coefficient which is computed as $\frac{|\mathcal{D}_n|}{|\mathcal{D}|}$ in FedAvg, where the $\mathcal{D} = \bigcup_{n=1}^{N} \mathcal{D}_n$. The empirical loss $\mathcal{L}_n(\theta_n)$ of the n-th client can be formulated as:

$$\mathcal{L}_n(\theta_n) = \mathbb{E}_{(x \in \mathcal{X}_n, y \in \mathcal{Y}_n) \sim \mathbb{P}_n(\mathcal{X}, \mathcal{Y})}[\ell(f_n(x; \theta_n), y)], \tag{2}$$

where $\ell(\cdot, \cdot)$ denotes the loss function on the clients. Following the local updates, the server $\mathcal{S}$ performs aggregation of multiple local models' parameters to derive the global model as $\theta_{\text{glob}} \leftarrow \sum_{n=1}^{N} z_n \theta_n$.

3.2 Problem Formulation in Federated UAD

As shown in Fig. 1, we consider a federated UAD setting with N independent clients that collaboratively perform a shared anomaly detection task using only normal samples during training and without sharing raw data. In this work, a client denotes an independent data holder, such as a factory site, production line, or screening center. The shared task refers to anomaly detection for the same inspection or screening target, e.g., PCB inspection, wafer defect detection, or chest X-ray screening. Consequently, the training dataset available at the n-th client is denoted as:

$$\mathcal{D}_n^{\text{train}} = (\mathcal{X}_n, \mathcal{Y}_n) \sim \mathbb{P}_n(\mathcal{X}), \forall y \in \mathcal{Y}_n, y = 0, \tag{3}$$

where $y = 0$ indicates that only normal samples are used for training. Although all the $\mathcal{D}_n^{\text{train}}$ are from the same anomaly detection task, normal data distribution in each client can be different due to manufacturing conditions or screening populations. We refer to this normal distribution heterogeneity as intra-class distribution bias in federated UAD, which can be formulated as:

$$\exists_{x_m \in \mathcal{D}_m^{\text{train}}, x_n \in \mathcal{D}_n^{\text{train}}} \mathbb{P}_m(x_m) \neq \mathbb{P}_n(x_n), \textit{while } m \neq n. \tag{4}$$

During testing, we evaluate the anomaly detection performance across all participating clients under the same federated UAD task. This evaluates whether the learned knowledge can generalize across heterogeneous client distributions within the shared task. Thus, the overall test dataset is defined as:

$$\begin{aligned} \mathcal{D}^{\text{test}} &= \bigcup_{n=1}^{N} \mathcal{D}_n^{\text{test}}, \textit{where} \\ \mathcal{D}_n^{\text{test}} &= (\mathcal{X}_n, \mathcal{Y}_n) \sim \mathbb{P}_n(\mathcal{X}, \mathcal{Y}), \exists y \in \mathcal{Y}_n, y = 1, \end{aligned} \tag{5}$$

where $y = 1$ denotes the anomalous samples in testing.

Different from supervised federated learning, federated UAD cannot directly formulate the local objective as supervised prediction from $\mathcal{X}_n \rightarrow \mathcal{Y}_n$. Thus, the empirical loss of the n-th client in UAD is formulated as:

$$\mathcal{L}_n(\theta_n) = \mathbb{E}_{(x \in \mathcal{X}_n) \sim \mathbb{P}_n(\mathcal{X})}[\ell(f_n(x; \theta_n))], \tag{6}$$

where $\ell(\cdot)$ denotes the local loss function for individual clients. It requires the n-th local model $f_n(\cdot; \theta_n)$ to approximate the normal sample distribution, i.e., $\mathbb{P}_n(\mathcal{X}^{\text{train}})$, under the condition of intra-class bias (refer to 4). For the federated UAD, shown in Fig. 1, the federated objective in one-class classification task is to generalize across all clients by constructing $\mathbb{P}(\{\mathcal{X}_n^{\text{train}}\}_{n=1}^{N})$. During the testing phase, the result of a sample $x^{\text{test}} \in \mathcal{D}^{\text{test}}$ can be expressed as follows:

$$\mathbb{1}(x^{\text{test}}) = \begin{cases} 0, & \text{if } f(x^{\text{test}}; \Omega) \sim \mathbb{P}(\{\mathcal{X}_n^{\text{train}}\}_{n=1}^{N}) \\ 1, & \text{otherwise,} \end{cases} \tag{7}$$

where Ω denotes the shared knowledge obtained through federated aggregation. For parameter-aggregation methods, Ω corresponds to the aggregated global parameter θ_{glob}. In FedDyMem, Ω denotes the global memory.

4 Methodology

4.1 Overview

As shown in Fig. 2, we propose an efficient federated learning framework with dynamic memory and memory-reduce for UAD, called FedDyMem. FedDyMem is designed for a shared federated UAD task, where participating clients use only normal samples from the same inspection or screening target but exhibit heterogeneous normal distributions. Instead of sharing raw data or full model parameters, FedDyMem performs collaborative anomaly detection by exchanging reduced memory banks between clients and the server. Before iterative training, an initial global memory bank is constructed by reducing memory features from clients and aggregating them on the server, which provides the reference memory for the first trainable round. In each communication round, FedDyMem follows three steps:

Step 1: Local training on clients. Each client receives the global memory bank from the previous round and updates its local projection layer and memory generator using only normal samples. The local objective aligns generated memory features with the shared global memory.

Step 2: Memory upload preparation. After local training, each client regenerates memory features from its normal samples and compresses them into a compact local memory bank through memory-reduce. Only this reduced memory bank is uploaded to the server.

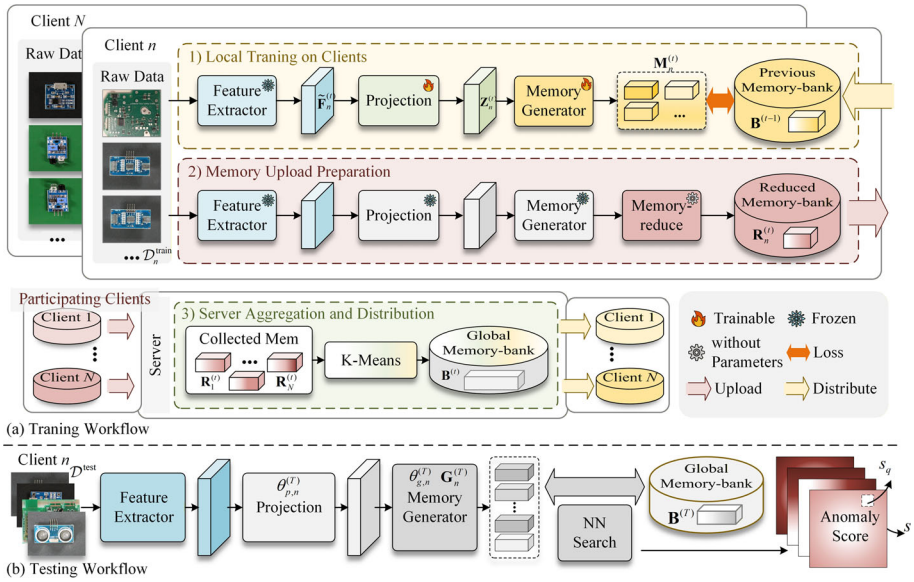

Fig. 2 Overview of our proposed FedDyMem framework

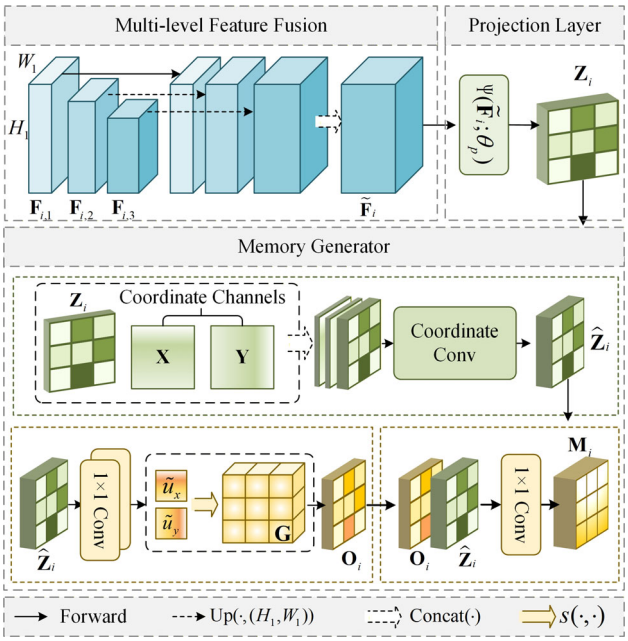

Fig. 3 Illustration of the memory feature generation process in each client model

Step 3: Server aggregation and distribution. The server aggregates the uploaded reduced memory banks into a new global memory bank and distributes it back to clients for the next round.

After training, each client performs anomaly detection using its locally trained modules and the final global memory bank. Following the anomaly score function in Roth et al. (2022), Lee et al. (2022), test memory features are compared with the global memory through nearest-neighbor search to compute anomaly scores.

4.2 Model Architecture in Clients

The client model consists of a frozen backbone, a projection layer, and a memory generator. Given an input image $x_{n,i}$, the client model extracts multi-level pretrained features and adapts them through the projection layer to generate a per-sample memory feature $\mathbf{M}_{n,i}$.

Frozen Backbone. Locally, FedDyMem employs a frozen CNN model $\Phi(\cdot)$ pre-trained on ImageNet to extract multi-level hierarchical features. Motivated by Roth et al. (2022), FedDyMem performs multi-level feature fusion on the extracted features. As shown in Fig. 3, for a given sample $x_{n,i} \in \mathbb{R}^{H\times W\times 3}$ which is from the n-th client's $\mathcal{D}_n$, the extracted features are hierarchically organized into L layers (e.g., $L = 5$ in the case of ResNet He et al. 2016). We define $\mathbf{F}_{n,i,l} \in \mathbb{R}^{H_l\times W_l\times C_l}$ as the output features of the l-th layer, obtained from $\Phi(x_{n,i})$, where i denotes the i-th sample from local dataset, l denotes the l-th layer and H_l, W_l, C_l are the size and channel of the feature map. In multi-level feature fusion, we employ a straightforward approach combining upsampling and feature concatenation to extract features from pre-trained CNNs. This process can be expressed as follows:

$$\tilde{\mathbf{F}}_{n,i} = \text{Concat}(\{\text{Up}(\mathbf{F}_{n,i,l}, (H_1, W_1))\}_{l=1}^{L}), \tag{8}$$

where $\text{Concat}(\cdot)$ represents the concatenation operation, $\text{Up}(\cdot, (H, W))$ denotes bilinear interpolation to the spatial dimensions (H, W) and (H_1, W_1) is the feature resolution of $l = 1$.

Projection Layer. Feature extractors pretrained on large-scale natural image datasets often exhibit significant distribution shifts when applied to industrial or medical images. To address this issue, we introduce a projection layer $\Psi(\cdot; \theta_{p,n})$ designed to adapt the extracted features to the anomaly detec-

tion domain. As shown in Fig. 3, the projection layer is implemented as a 1×1 convolutional layer followed by an activation function. The output of $\Psi(\tilde{\mathbf{F}}_{n,i}; \theta_{p,n})$ is denoted as $\mathbf{Z}_{n,i}$.

Memory Generator. The memory bank-based approach (Cohen & Hoshen, 2020; Defard et al., 2021; Roth et al., 2022) has demonstrated significant performance in UAD. However, the memory banks utilized in these existing methods are non-trainable, meaning they remain unchanged following initialization. Due to the local bias in feature distributions for federated UAD, using the static memory bank will lead to overfitting on domain-specific data, increasing the risk of error. Furthermore, existing methods operate mainly on a discrete feature space, which increases the variations in feature distributions between clients for normal samples (Lee et al., 2025). As shown in Fig. 3, we propose a memory generator that integrates spatial information and ensures the continuity of feature space. The memory generator $\Gamma(\cdot; \theta_{g,n}, \mathbf{G}_n)$ utilizes a coordinate convolution layer to encode spatial location information by incorporating additional coordinate channels, thereby facilitating the network's ability to learn spatial transformations more effectively (Liu et al., 2018). The output of coordinate convolution is represented as $\hat{\mathbf{Z}}_{n,i}$ using the parameter $\theta_{g,n}$.

Then, to enhance the continuity of the memory bank, the memory generator utilizes a grid-based approach to construct a continuous feature space. We define a trainable grid space as $\mathbf{G}_n \in \mathbb{R}^{H_g \times W_g \times C}$, where H_g and W_g are hyperparameters indicating the size of the continuous space. The memory generator maps $\hat{\mathbf{Z}}_{n,i}$ to pixel-wise coordinates $\mathbf{U}_{n,i} \in \mathbb{R}^{H_1 \times W_1 \times 2}$ through two 1×1 convolutional layers in our memory generator. A sample function $s(\cdot, \cdot)$ is employed to extract features from the continuous space $\mathbf{G}_n$ at specific coordinate values. For each spatial location (h, w) in $\mathbf{U}_{n,i}$, we denote the predicted coordinate as $(u_x, u_y) = \mathbf{U}_{n,i}[h, w]$. The normalized coordinates are computed as follows:

$$\tilde{u}_x = \frac{(u_x + 1)}{2} \cdot (W_g - 1), \quad \tilde{u}_y = \frac{(u_y + 1)}{2} \cdot (H_g - 1), \quad (9)$$

and the result of $s(\mathbf{G}_n, \mathbf{U}_{n,i})$ is defined as $\mathbf{O}_{n,i}$. Each pixel value in $\mathbf{O}_{n,i}$ can be sampled as:

$$\mathbf{O}_{n,i,h,w} = \sum_{a=1}^{2} \sum_{b=1}^{2} \omega_{ab} \mathbf{G}_{n \lfloor \tilde{u}_y \rfloor + (a-1), \lfloor \tilde{u}_x \rfloor + (b-1)}, \quad (10)$$

where

$$\omega_{ab} = (1 - |\tilde{u}_x - (\lfloor \tilde{u}_x \rfloor + (b-1))|) \\ (1 - |\tilde{u}_y - (\lfloor \tilde{u}_y \rfloor + (a-1))|),$$

and $\lfloor \cdot \rfloor$ represents the floor operation. Finally, the features $\mathbf{O}_{n,i}$, sampled from the continuous feature space, are concatenated with the coordinate convolution outputs $\hat{\mathbf{Z}}_{n,i}$. This concatenated tensor is then passed through a convolutional layer to generate the memory feature $\mathbf{M}_{n,i} \in \mathbb{R}^{H_1 \times W_1 \times C}$ for client n.

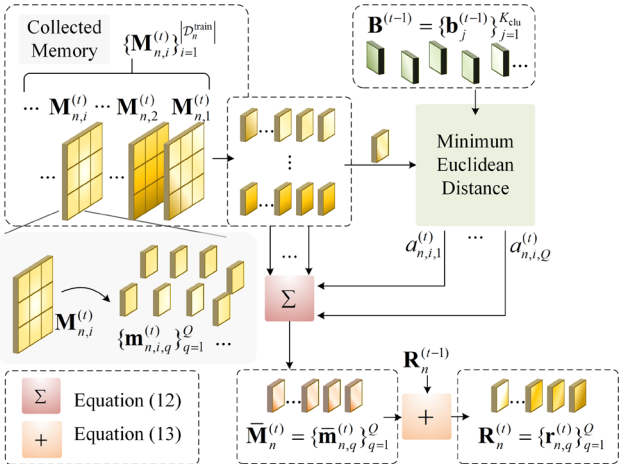

Fig. 4 Illustration of Memory-reduce

4.3 Memory-Reduce Before Uploading

The earlier memory-based methods (Cohen & Hoshen, 2020; Roth et al., 2022) involved storing all sample memory features from the training dataset within the memory bank. Accordingly, the capacity of the memory bank scales proportionally with the size of the local dataset. For federated UAD, this design leads to high communication overhead when applied to large-scale datasets and also introduces privacy risks. When each memory feature corresponds directly to a raw input sample, it becomes vulnerable to privacy leakage under reconstruction or inversion attacks. Moreover, the varying sizes of memory banks also introduce challenges in the aggregation process. Motivated by this, FedDyMem incorporates a memory-reduce method to compress the collected memory features from the local dataset, thereby reducing the mutual information between the memory features and the raw data.

Motivated by this, after local training at the t-th communication round, FedDyMem applies memory-reduce on each client to compress the collected memory features $\{\mathbf{M}_{n,i}^{(t)}\}_{i=1}^{|\mathcal{D}_n^{\text{train}}|} \in \mathbb{R}^{|\mathcal{D}_n^{\text{train}}| \times H_1 \times W_1 \times C}$ of all samples into a fixed-size reduced local memory bank $\mathbf{R}_n^{(t)}$. Only the reduced memory bank is uploaded to the server for subsequent aggregation.

To address the memory bank size issue, Lee et al. (2022) employs an Exponential Moving Average (EMA) method. However, the reduced memory bank remains sensitive to the sequence in which input samples are processed. Thus, we introduce a dynamic weighted average that optimizes mem-

ory size while preserving sample order independence and effectively correlating with the number of communication rounds. As shown in Fig. 4, we apply the dynamic weighted averaging operation to compute the $\bar{\mathbf{M}}_n^{(t)}$. Let $Q = H_1 W_1$ denote the number of patch features in each local memory feature and $\mathbf{M}_{n,i}^{(t)} = \{\mathbf{m}_{n,i,q}^{(t)}\}_{q=1}^{Q}$. The global memory contains K_{clu} prototypes, denoted as $\mathbf{B}^{(t-1)} = \{\mathbf{b}_j^{(t-1)}\}_{j=1}^{K_{\text{clu}}}$ (detailed in Sect. 4.4. The dynamic weight for q-th patch feature is computed by the nearest distance:

$$a_{n,i,q}^{(t)} = \min_{1 \le j \le K_{\text{clu}}} \text{dis}\left(\mathbf{m}_{n,i,q}^{(t)}, \mathbf{b}_j^{(t-1)}\right), \tag{11}$$

where the $\text{dis}(\cdot,\cdot)$ denotes the Euclidean distance. The average memory is denoted as $\bar{\mathbf{M}}_n^{(t)} = \{\bar{\mathbf{m}}_{n,q}^{(t)}\}_{q=1}^{Q}$, where each average patch feature is computed by:

$$\bar{\mathbf{m}}_{n,q}^{(t)} = \sum_{i=1}^{|\mathcal{D}_n^{\text{train}}|} a_{n,i,q}^{(t)} \mathbf{m}_{n,i,q}^{(t)} \Big/ \sum_{i=1}^{|\mathcal{D}_n^{\text{train}}|} a_{n,i,q}^{(t)}, \tag{12}$$

Thus, the average feature $\bar{\mathbf{M}}_n^{(t)}$ is influenced by the distance between the memory representation $\mathbf{M}_n^{(t)}$ and the previous memory bank $\mathbf{B}^{(t-1)}$.

After computing $\bar{\mathbf{M}}_n^{(t)}$, a round-based EMA (rEMA) method is employed to update the memory bank. Each client keeps a local cache of its previously uploaded reduced memory bank $\mathbf{R}_n^{(t-1)}$, which is used as the historical term in the rEMA update. This approach ensures smoother and more stable updates during the training process. The update rule is defined as:

$$\mathbf{R}_n^{(t)} = \alpha_t \bar{\mathbf{M}}_n^{(t)} + (1 - \alpha_t)\mathbf{R}_n^{(t-1)}, \tag{13}$$

where α_t represents the exponential decay rate, dynamically calculated as $\alpha_t = 1/(t+1)$.

4.4 Memory Aggregation on Server

At round t, the server $\mathcal{S}$ collects the reduced memory banks from N clients, denoted as $\{\mathbf{R}_n^{(t)}\}_{n=1}^{N}$. The objective of the server is to aggregate these memory banks into a unified, representative memory bank $\mathbf{B}^{(t)}$. Representation-based federated learning approaches, such as those proposed in Tan et al. (2022a), Tan et al. (2022b), utilize weighted aggregation of class prototypes during model updates. However, significant feature bias in the early training stages leads to aggregated memory that deviates significantly from the client-specific memory, which often contains incomplete product types. This misalignment hinders convergence, posing a challenge for clients with limited data diversity. Thus, we employ a clustering-based approach to aggregate the collected memory banks effectively.

Let $Q = H_1 W_1$ denote the number of local reduced prototypes in each uploaded memory bank. Therefore, the server clusters only NQ uploaded prototypes. Each reduced memory is flattened into a set of Q patch-level prototypes, denoted as $\mathbf{R}_n^{(t)} = \{\mathbf{r}_{n,q}^{(t)} \in \mathbb{R}^C\}_{q=1}^{Q}$. The server then collects all uploaded prototypes as:

$$\mathcal{P}^{(t)} = \bigcup_{n=1}^{N} \{\mathbf{r}_{n,q}^{(t)}\}_{q=1}^{Q}. \tag{14}$$

FedDyMem applies k-means clustering to $\mathcal{P}^{(t)}$ and uses the cluster centers as global memory prototypes:

$$\mathbf{B}^{(t)} = \{\mathbf{b}_j^{(t)} \in \mathbb{R}^C\}_{j=1}^{K_{\text{clu}}} = \text{Kmeans}(\mathcal{P}^{(t)}, K_{\text{clu}}), \tag{15}$$

where K_{clu} denotes the number of global prototypes after aggregation. The clustering results describe the overall distribution and similarity relationships among patch-level prototypes stored across all reduced local memory banks. By maintaining multiple representative centers, the aggregated memory bank can cover different normal variations across clients more effectively than a single averaged representation.

Since clustering is performed on the reduced memory banks, the server processes NQ uploaded prototypes in each round. The computational complexity of k-means aggregation is $\mathcal{O}(I_{\text{clu}} NQ K_{\text{clu}} C)$, where I_{clu} is the number of k-means iterations.

4.5 Training Pipeline

Initialization. The initialization stage aims to construct the first shared reference memory bank $\mathbf{B}^{(0)}$ before any iterative local optimization is performed. We denote this stage as round $t = 0$. Different from the subsequent training rounds, no metric loss is computed at $t = 0$, and the trainable projection layer and memory generator are not updated in this stage.

To ensure that the initial memory features are represented in a consistent feature space, all clients start from the same initialization. Specifically, each client employs the same frozen backbone and starts with identical projection parameters $\theta_p^{(0)}$, memory generator parameters $\theta_g^{(0)}$, and grid space $\mathbf{G}^{(0)}$. For each normal training sample $x_{n,i} \in \mathcal{D}_n^{\text{train}}$, client n first extracts multi-level features using the frozen pretrained backbone $\Phi(\cdot)$ and obtains the fused feature $\tilde{\mathbf{F}}_{n,i}^{(0)}$. The initial per-sample memory feature $\mathbf{M}_{n,i}^{(0)}$ is then generated by the initialized projection layer and memory generator, denoted as $\mathbf{M}_{n,i}^{(0)} = \Gamma(\Psi(\tilde{\mathbf{F}}_{n,i}^{(0)}; \theta_{p,n}^{(0)}); \theta_{g,n}^{(0)}, \mathbf{G}_n^{(0)})$.

After obtaining memory features, each client applies memory-reduce to construct local memory bank $\mathbf{R}_n^{(0)}$. Since no previous global memory bank exists at $t = 0$, the dynamic

Input: Training sets $\{\mathcal{D}_n^{\text{train}}\}_{n=1}^N$, server $\mathcal{S}$, number of global prototypes K_{clu}.
Output: Initial global memory bank $\mathbf{B}^{(0)}$, initial reduced local memories $\{\mathbf{R}_n^{(0)}\}_{n=1}^N$, and initialized local modules $\{\theta_{p,n}^{(0)}, \theta_{g,n}^{(0)}, \mathbf{G}_n^{(0)}\}_{n=1}^N$.
Initialize $(\theta_{p,n}^{(0)}, \theta_{g,n}^{(0)}, \mathbf{G}_n^{(0)})$ for each client with the same initialization $(\theta_p^{(0)}, \theta_g^{(0)}, \mathbf{G}^{(0)})$.
for each client $n = 1, \ldots, N$ **in parallel do**
  for $i = 1$ to $\left|\mathcal{D}_n^{\text{train}}\right|$ **do**
    $\{\mathbf{F}_{n,i,l}^{(0)}\}_{l=1}^L \leftarrow \Phi(x_{n,i})$;
    $\tilde{\mathbf{F}}_{n,i}^{(0)} \leftarrow$ Compute by Equation (8);
    $\mathbf{M}_{n,i}^{(0)} \leftarrow \Gamma\left(\Psi\left(\tilde{\mathbf{F}}_{n,i}^{(0)}; \theta_{p,n}^{(0)}\right); \theta_{g,n}^{(0)}, \mathbf{G}_n^{(0)}\right)$;
  end for
  $\mathbf{R}_n^{(0)} \leftarrow$ Compute the initial reduced local memory bank by Equation (16);
  Upload $\mathbf{R}_n^{(0)}$ to the server $\mathcal{S}$;
end for
Construct $\mathcal{P}^{(0)} = \bigcup_{n=1}^N \{\mathbf{r}_{n,q}^{(0)}\}_{q=1}^Q$ by flattening $\{\mathbf{R}_n^{(0)}\}_{n=1}^N$ on $\mathcal{S}$;
$\mathbf{B}^{(0)} \leftarrow \text{Kmeans}\left(\mathcal{P}^{(0)}, K_{\text{clu}}\right)$;
return $\mathbf{B}^{(0)}, \{\mathbf{R}_n^{(0)}\}_{n=1}^N, \{\theta_{p,n}^{(0)}, \theta_{g,n}^{(0)}, \mathbf{G}_n^{(0)}\}_{n=1}^N$.

Algorithm 1: Initialization of FedDyMem at Round $t = 0$.

weights are set as $a_{n,i,q}^{(0)} = 1$, and the rEMA update is not applied. Thus, the reduced local memory bank is computed as:

$$\mathbf{R}_n^{(0)} = \frac{1}{\left|\mathcal{D}_n^{\text{train}}\right|} \sum_{i=1}^{\left|\mathcal{D}_n^{\text{train}}\right|} \mathbf{M}_{n,i}^{(0)}. \tag{16}$$

The server collects the reduced memory banks from all clients and constructs the initial global memory bank $\mathbf{B}^{(0)}$ through the aggregation described in Sect. 4.4. Because all clients share the same initial projection, memory-generator, and grid parameters, the initial memory features are not arbitrary random targets. Conditioned on this shared initialization, they are deterministic, data-dependent transformations of the frozen pretrained backbone features in a common feature coordinate system. After memory-reduce and server-side K-means aggregation, the initial global memory bank $\mathbf{B}^{(0)}$ summarizes the aggregated initial representations of normal training data across clients. It is then distributed as the fixed reference memory for the first trainable round ($t = 1$). The initialization is summarized in Algorithm 1.

Local Training. After initialization, FedDyMem performs iterative local training. For each trainable communication round $t \geq 1$, client n receives the global memory bank $\mathbf{B}^{(t-1)}$ from the server. Given a normal sample $x_{n,i} \in \mathcal{D}_n^{\text{train}}$, client n generates the current memory feature $\mathbf{M}_{n,i}^{(t)}$ by the backbone, the projection layer and the memory generator. During this process, the pretrained backbone $\Phi(\cdot)$ is frozen. The trainable components are the projection layer parameters $\theta_{p,n}^{(t)}$, the memory generator parameters $\theta_{g,n}^{(t)}$, and the trainable grid space $\mathbf{G}_n^{(t)}$.

Considering the bias in feature distributions across clients, we introduce a metric loss to optimize the local model to produce normal features that are more closely aligned with the shared global memory bank. For each local memory feature, we use $Q = H_1 W_1$ as the number of patch features. Thus, the memory feature $\mathbf{M}_{n,i}^{(t)}$ can be denoted as $\{\mathbf{m}_{n,i,q}^{(t)}\}_{q=1}^Q$ and the reference global memory $\mathbf{B}^{(t-1)}$ can be represented as $\{\mathbf{b}_j^{(t-1)}\}_{j=1}^{K_{\text{clu}}}$. For each local patch feature $\mathbf{m}_{n,i,q}^{(t)}$, its k_{nn} nearest neighbors are retrieved from the K_{clu} global prototypes in $\mathbf{B}^{(t-1)}$. We denote its r-th nearest neighbor as $\mathbf{b}_{n,i,q,r}^{(t-1)}$ and the metric loss is then defined as:

$$\begin{aligned} &\ell(\mathbf{M}_{n,i}^{(t)}, \mathbf{B}^{(t-1)}) \\ &= \frac{1}{Qk_{\text{nn}}} \sum_{q=1}^{Q} \sum_{r=1}^{k_{\text{nn}}} \max(0, \text{dis}(\mathbf{m}_{n,i,q}^{(t)}, \mathbf{b}_{n,i,q,r}^{(t-1)}) - \tau_\ell), \end{aligned} \tag{17}$$

where k_{nn} is the number of nearest neighbors, and τ_ℓ is the margin threshold. The local objective of client n at round t is then defined as:

$$\mathcal{L}_n^{(t)} = \frac{1}{\left|\mathcal{D}_n^{\text{train}}\right|} \sum_{i=1}^{\left|\mathcal{D}_n^{\text{train}}\right|} \ell(\mathbf{M}_{n,i}^{(t)}, \mathbf{B}^{(t-1)}). \tag{18}$$

Communication. After E local training epochs at communication round $t \geq 1$, FedDyMem performs memory-based communication between clients and the server. The objective of this step is to update the shared global memory bank without exchanging local model parameters or gradients.

For client n, after local optimization is completed, the updated projection layer and memory generator are kept fixed and used to regenerate the memory features of all normal training samples, denoted as $\{\mathbf{M}_{n,i}^{(t)}\}_{i=1}^{\left|\mathcal{D}_n^{\text{train}}\right|}$. These memory features are not directly uploaded. Instead, client n applies the memory-reduce defined in Sect. 4.3 to obtain the local memory bank $\mathbf{R}_n^{(t)}$.

After receiving the uploaded reduced memory banks, the server aggregates them into a new global memory bank $\mathbf{B}^{(t)}$ using the aggregation described in Sect. 4.4. The updated global memory bank $\mathbf{B}^{(t)}$ is then distributed to all clients and used as the fixed reference memory for the next round of local training. The complete training and communication protocol is summarized in Algorithm 2.

4.6 Testing and Anomaly Scoring

After the final communication round T, FedDyMem performs inference using local model trained on each client

Input: Training sets $\{\mathcal{D}_n^{\text{train}}\}_{n=1}^N$, server $\mathcal{S}$, communication rounds T, local epochs E, number of nearest neighbors k_{nn}; loss threshold τ_ℓ; number of global prototypes K_{clu}.
Output: Final global memory bank $\mathbf{B}^{(T)}$ and trained local modules $\{\theta_{p,n}^{(T)}, \theta_{g,n}^{(T)}, \mathbf{G}_n^{(T)}\}_{n=1}^N$.

$\mathbf{B}^{(0)}, \{\mathbf{R}_n^{(0)}\}_{n=1}^N, \{\theta_{p,n}^{(0)}, \theta_{g,n}^{(0)}, \mathbf{G}_n^{(0)}\}_{n=1}^N \leftarrow$ Obtain by Algorithm 1;
for $t = 1$ to T **do**
  for each client **in parallel do**
    $(\theta_{p,n}^{(t)}, \theta_{g,n}^{(t)}, \mathbf{G}_n^{(t)}) \leftarrow$ UPDATE$(n, t, E, \mathcal{D}_n^{\text{train}}, \mathbf{B}^{(t-1)})$;
    $\mathbf{R}_n^{(t)} \leftarrow$ REDUCE$(n, t, \mathcal{D}_n^{\text{train}}, \mathbf{B}^{(t-1)}, \mathbf{R}_n^{(t-1)}, \theta_{p,n}^{(t)}, \theta_{g,n}^{(t)}, \mathbf{G}_n^{(t)})$;
    Upload $\mathbf{R}_n^{(t)}$ to the server $\mathcal{S}$;
  end for
  Server $\mathcal{S}$ collects uploaded reduced memory banks $\{\mathbf{R}_n^{(t)}\}_{n=1}^N$;
  Construct $\mathcal{P}^{(t)}$ by Equation (14);
  $\mathbf{B}^{(t)} \leftarrow \text{Kmeans}(\mathcal{P}^{(t)}, K_{\text{clu}})$;
  Distribute $\mathbf{B}^{(t)}$ to all participating clients for the next round;
end for
Distribute the final global memory bank $\mathbf{B}^{(T)}$ to all clients.
return $\mathbf{B}^{(T)}, \{\theta_{p,n}^{(T)}, \theta_{g,n}^{(T)}, \mathbf{G}_n^{(T)}\}_{n=1}^N$

UPDATE$(n, t, E, \mathcal{D}_n^{\text{train}}, \mathbf{B}^{(t-1)})$
for $e = 1$ to E **do**
  for $i = 1$ to $|\mathcal{D}_n^{\text{train}}|$ **do**
    $\mathbf{M}_{n,i}^{(t)} \leftarrow$ Generate memory feature by $\Phi(\cdot), \Psi(\cdot; \theta_{p,n}^{(t)}), \Gamma(\cdot; \theta_{g,n}^{(t)}, \mathbf{G}_n^{(t)})$;
    $\ell(\mathbf{M}_{n,i}^{(t)}, \mathbf{B}^{(t-1)}) \leftarrow$ Compute the local metric loss by Equation (17);
    Update $\theta_{p,n}^{(t)}$, $\theta_{g,n}^{(t)}$, and $\mathbf{G}_n^{(t)}$ by backpropagation, while keeping $\Phi(\cdot)$ frozen;
  end for
end for
return $\theta_{p,n}^{(t)}, \theta_{g,n}^{(t)}, \mathbf{G}_n^{(t)}$

REDUCE$(n, t, \mathcal{D}_n^{\text{train}}, \mathbf{B}^{(t-1)}, \mathbf{R}_n^{(t-1)}, \theta_{p,n}^{(t)}, \theta_{g,n}^{(t)}, \mathbf{G}_n^{(t)})$
for $i = 1$ to $|\mathcal{D}_n^{\text{train}}|$ **do**
  $\mathbf{M}_{n,i}^{(t)} \leftarrow$ Regenerate using the frozen $\Phi(\cdot)$ and the updated $\Psi(\cdot; \theta_{p,n}^{(t)})$, $\Gamma(\cdot; \theta_{g,n}^{(t)}, \mathbf{G}_n^{(t)})$;
end for
$\alpha_t \leftarrow 1/(t+1)$;
$\mathbf{R}_n^{(t)} \leftarrow$ Compute the reduced memory by Equations (11)–(13) with $\{\mathbf{M}_{n,i}^{(t)}\}_{i=1}^{|\mathcal{D}_n^{\text{train}}|}, \mathbf{B}^{(t-1)}, \mathbf{R}_n^{(t-1)}, \alpha_t$;
return $\mathbf{R}_n^{(t)}$

Algorithm 2: FedDyMem Training and Memory Communication.

together with a shared global memory bank, rather than using a single global anomaly detector obtained by parameter aggregation. As shown in Fig. 2b, each client receives the globally aggregated memory bank $\mathbf{B}^{(T)}$ from the server at the end of training. Following prior works Roth et al. (2022), Lee et al. (2022), we utilize a nearest neighbor search in the feature space to calculate the anomaly score for each test sample in $\mathcal{D}^{\text{test}}$.

Given a test image x^{test} on n-th client, the well trained projection layer $\Psi(\cdot; \theta_{p,n}^{(T)})$ and memory generator $\Gamma(\cdot; \theta_{g,n}^{(T)}, \mathbf{G}_n^{(T)})$ are used to extract the memory feature representation $\mathbf{M}^{\text{test}}$ following the same procedure as in training. For each spatial location, the corresponding feature patch $\mathbf{m}_q^{\text{test}} \in \mathbb{R}^C$ is compared with the final global memory bank $\mathbf{B}^{(T)}$ to compute its anomaly score. The global memory bank is represented as an unordered set of normal prototypes $\{\mathbf{b}_j^{(T)} \in \mathbb{R}^C\}_{j=1}^{K_{\text{clu}}}$. Following previous works, the patch-level anomaly score s_q is computed as the minimum Euclidean distance to the nearest global prototype $\min_{1 \le j \le K_{\text{clu}}} \text{dis}(\mathbf{m}_q^{\text{test}}, \mathbf{b}_j^{(T)})$. For anomaly localization, the patch-level scores are upsampled to the original image resolution to obtain the anomaly map. The image-level anomaly score is computed by the softmax-weighted maximum strategy:

$$s = \max_{1 \le q \le Q} \left(s_q \cdot \frac{\exp(s_q)}{\sum_{q'=1}^{Q} \exp(s_{q'})} \right). \tag{19}$$

4.7 Privacy Analysis

In federated learning, Mutual Information(MI) is a fundamental tool for evaluating privacy. Prior studies Zhang et al. (2023), Alsulaimawi (2024), Tan et al. (2024) indicate that MI provides a principled paradigm for privacy analysis and can jointly improve privacy and utility. Consistent with prior work, we adopt an MI–based methodology to analyze FedDyMem–s privacy. For two statistics X and Y, their mutual information $I(X; Y)$ quantifies statistical dependence; under Gaussian assumptions,

$$I(X; Y) = -\tfrac{1}{2} \log\bigl(1 - \rho_{xy}^2\bigr). \tag{20}$$

where ρ_{xy} denotes the Pearson correlation coefficient. A smaller $I(X; Y)$ implies that an adversary can extract less information about X from observing Y, indicating stronger privacy.

To make the following analysis well-defined for tensor-valued images and memory banks, we consider one fixed scalar statistic at a time. Specifically, for a fixed client n, round t, patch q, and channel c, let $X_j = \phi(x_{n,j})$ be a scalar statistic of the j-th raw sample and let $Y_i = [\mathbf{m}_{n,i,q}^{(t)}]_c$ be the

corresponding scalar memory statistic. The result below is therefore a conditional, per-coordinate dependence analysis.

Let $D_n = |\mathcal{D}_n^{\text{train}}|$ and, for a non-degenerate coordinate with $\sum_{i=1}^{D_n} a_{n,i,q}^{(t)} > 0$, define $\tilde{w}_i = a_{n,i,q}^{(t)} / \sum_{r=1}^{D_n} a_{n,r,q}^{(t)}$ and $\bar{Y} = \sum_{i=1}^{D_n} \tilde{w}_i Y_i$. Let $\mathcal{H}^{(t)}$ denote the information fixed for the current reduction, including the realized normalized weights, the previous reduced memory, the previous global memory, and the local encoder state. By Eq. (13), the transmitted coordinate is $Y_{\text{red}} = [\mathbf{R}_n^{(t)}]_{q,c} = \alpha_t \bar{Y} + (1-\alpha_t) Y_{\text{hist}}$, where $Y_{\text{hist}} = [\mathbf{R}_n^{(t-1)}]_{q,c}$ is fixed conditional on $\mathcal{H}^{(t)}$; at $t = 0$, $Y_{\text{red}} = \bar{Y}$. Conditional on $\mathcal{H}^{(t)}$, memory-reduce is a deterministic post-processing of $\{Y_i\}_{i=1}^{D_n}$, and DPI gives

$$I\left(X_j; Y_{\text{red}} \mid \mathcal{H}^{(t)}\right) \le I\left(X_j; \{Y_i\}_{i=1}^{D_n} \mid \mathcal{H}^{(t)}\right). \tag{21}$$

The following lemma gives a quantitative comparison with the corresponding per-sample scalar memory statistic under the stated distributional assumptions.

Lemma 1 (Correlation quantitative bound) *Let $\{Y_i\}_{i=1}^N$ be scalar statistics and define the normalized weighted average $Y = \sum_{i=1}^N \tilde{w}_i Y_i$ with $\tilde{w}_i \ge 0$, $\sum_i \tilde{w}_i = 1$. All moments, correlations, and independence statements below are conditional on $\mathcal{H}^{(t)}$; this conditioning is omitted in the notation. The normalized weights are thus treated as fixed in the bound. Assume:*

1. *X_j is independent of Y_i for $i \ne j$.*
1. *Each Y_i has finite, strictly positive variance $\sigma_{Y_i}^2$, and $\text{Cov}(Y_i, Y_k) \ge 0$ for $i \ne k$.*

Denote $\sigma_{Y_i}^2 = \text{Var}(Y_i)$, then

$$|\rho(X_j, Y)| \le \frac{\tilde{w}_j \sigma_{Y_j}}{\sqrt{\sum_{i=1}^N \tilde{w}_i^2 \sigma_{Y_i}^2}} |\rho(X_j, Y_j)|. \tag{22}$$

Proof 1 By linearity and independence, $\text{Cov}(X_j, Y) = \tilde{w}_j \text{Cov}(X_j, Y_j)$, hence

$$|\rho(X_j, Y)| = \frac{|\text{Cov}(X_j, Y)|}{\sigma_{X_j} \sigma_Y} = \frac{\tilde{w}_j |\text{Cov}(X_j, Y_j)|}{\sigma_{X_j} \sigma_Y}. \tag{23}$$

For the denominator,

$$\text{Var}(Y) = \sum_{i=1}^N \tilde{w}_i^2 \sigma_{Y_i}^2 + 2 \sum_{1 \le i < k \le N} \tilde{w}_i \tilde{w}_k \text{Cov}(Y_i, Y_k). \tag{24}$$

Since $\text{Cov}(Y_i, Y_k) \ge 0$, with $\sigma_Y = \sqrt{\text{Var}(Y)}$, the variance satisfies $\sigma_Y = \sqrt{\text{Var}(Y)} \ge \sqrt{\sum_{i=1}^N \tilde{w}_i^2 \sigma_{Y_i}^2}$.

Substituting this bound into the denominator of (23) with $|\text{Cov}(X_j, Y_j)| = \sigma_{X_j} \sigma_{Y_j} |\rho(X_j, Y_j)|$, completes the proof. □

Theorem 1 (Memory-reduce privacy) *Under the conditional Gaussian approximation used above and the conditions of Lemma 1, the reduced scalar statistic satisfies*

$$I\left(X_j; Y_{\text{red}} \mid \mathcal{H}^{(t)}\right) \le I\left(X_j; Y_j \mid \mathcal{H}^{(t)}\right). \tag{25}$$

The inequality is strict whenever $0 < |\rho(X_j, Y_j \mid \mathcal{H}^{(t)})| < 1$ and the target sample does not provide the only nonzero effective contribution to the weighted average.

Proof 2 For the fixed (n, t, q, c), Eq. (13) gives $Y_{\text{red}} = \alpha_t \bar{Y} + (1-\alpha_t) Y_{\text{hist}}$. Since Y_{hist} is fixed conditional on $\mathcal{H}^{(t)}$ and $\alpha_t > 0$, $|\rho(X_j, Y_{\text{red}} \mid \mathcal{H}^{(t)})| = |\rho(X_j, \bar{Y} \mid \mathcal{H}^{(t)})|$. Applying Lemma 1 to $\bar{Y}$ yields

$$\left|\rho\left(X_j, Y_{\text{red}} \mid \mathcal{H}^{(t)}\right)\right| \le \gamma_j \left|\rho\left(X_j, Y_j \mid \mathcal{H}^{(t)}\right)\right|, \tag{26}$$

where

$$\gamma_j^2 = \frac{\tilde{w}_j^2 \sigma_{Y_j}^2}{\sum_{i=1}^{D_n} \tilde{w}_i^2 \sigma_{Y_i}^2} \le 1. \tag{27}$$

Here, $\gamma_j < 1$ whenever at least one other sample has nonzero normalized weight and positive variance. When $0 < |\rho(X_j, Y_j \mid \mathcal{H}^{(t)})| < 1$, the conditional Gaussian MI formula is monotone in $|\rho|$, and therefore

$$\frac{I\left(X_j; Y_{\text{red}} \mid \mathcal{H}^{(t)}\right)}{I\left(X_j; Y_j \mid \mathcal{H}^{(t)}\right)} \le \frac{-\frac{1}{2}\log\left(1 - \gamma_j^2 \rho\left(X_j, Y_j \mid \mathcal{H}^{(t)}\right)^2\right)}{-\frac{1}{2}\log\left(1 - \rho\left(X_j, Y_j \mid \mathcal{H}^{(t)}\right)^2\right)}. \tag{28}$$

The right-hand side is no greater than 1 because $0 \le \gamma_j \le 1$, and it is strictly smaller than 1 under the condition above. The zero-correlation case is immediate. This proves the stated conditional bound. □

For an asymptotic statement, an additional non-dominance condition is required: $\gamma_j^2 \to 0$ as the number of contributing samples grows. This condition is satisfied, for example, when normalized weights remain approximately balanced and the conditional variances are bounded above and away from zero. Under this condition and the assumptions above, $|\rho(X_j, Y_{\text{red}} \mid \mathcal{H}^{(t)})| \to 0$ and $I(X_j; Y_{\text{red}} \mid \mathcal{H}^{(t)}) \to 0$. Thus, memory-reduce attenuates the conditional per-sample dependence whenever no single sample dominates the weighted aggregate. Furthermore, FedDyMem exchanges statistical summaries (memory banks) rather than model parameters (Tan et al., 2022a), while the reconstruction, membership, and client-inference attacks in Sect. 6.7 provide complementary empirical evidence of reduced leakage.

5 Convergence Analysis

We analyze the convergence of FedDyMem under a non-convex loss. Let the loss at local iteration e in communication round t be $\mathcal{L}_e^{(t)} := \mathbb{E}_{x\sim\mathcal{D}_n}\big[\ell(\mathbf{M}_e^{(t)},\ \mathbf{B}^{(t-1)})\big]$, where $\mathbf{M}_e^{(t)} = f(x;\theta_e^{(t)})$ and its Jacobian as $J_f(x;\theta_e^{(t)})$. Here $\mathbf{B}^{(t-1)}$ is the center memory received at round t. Some common assumptions based on existing frameworks are reasonably presented below.

Assumption 1 (L_M*-smooth*) Within a fixed communication round t, treat $\mathbf{B}^{(t-1)}$ as a constant. Excluding the measure-zero non-differentiable set,

$$\|\nabla_{\mathbf{M}}\ell(\mathbf{M},\mathbf{B}) - \nabla_{\mathbf{M}}\ell(\tilde{\mathbf{M}},\mathbf{B})\|_2 \le L_M\|\mathbf{M}-\tilde{\mathbf{M}}\|_2, \tag{29}$$

and $\|\nabla_{\mathbf{M}}\ell(\mathbf{M},\mathbf{B})\|_2 \le G_M$. By the chain rule, the composite smoothness constant satisfies $L_\theta \le B_J^2 L_M$, and $\|\nabla_\theta\ell(\mathbf{M},\mathbf{B})\|_2 \le B_J G_M$.

Assumption 2 *(Bounded features and Jacobian)* There exist constants $R_M, B_J > 0$ such that, for any θ, x, satisfies $\|f(x;\theta)\|_2 \le R_M$, $\|J_f(x;\theta)\|_2 \le B_J$.

Assumption 3 *(Unbiased stochastic gradients with bounded variance)* The local SGD gradient estimate $g_e^{(t)}$ satisfies

$$\mathbb{E}\big[g_e^{(t)}\big] = \nabla_\theta\,\mathbb{E}_{x\sim\mathcal{D}_n}\big[\ell(\mathbf{M}_e^{(t)},\mathbf{B}^{(t-1)})\big], \tag{30}$$

and $\mathbb{E}\big\|g_e^{(t)} - \nabla_\theta\mathbb{E}_x[\ell(\mathbf{M}_e^{(t)},\mathbf{B}^{(t-1)})]\big\|_2^2 \le \sigma^2$.

Assumption 4 (*Lipschitz continuity in* $\mathbf{B}$) Client-side reduction and server aggregation are convex combinations. Together with Assumption 2, this implies $\|\mathbf{B}^{(t)}\|_2 \le R_M$, hence $\mathrm{dis}(\mathbf{m}_q,\mathbf{b}_k) \le 2R_M$ and $0 \le \ell(\cdot,\cdot) \le 2R_M$. Moreover, the map $\ell(\mathbf{M},\cdot)$ is 1-Lipschitz in $\mathbf{B}$:

$$\big|\ell(\mathbf{M},\mathbf{B}) - \ell(\mathbf{M},\tilde{\mathbf{B}})\big| \le \|\mathbf{B}-\tilde{\mathbf{B}}\|_2. \tag{31}$$

This follows from the bounded convex hull, max-hinge, and the triangle inequality. Moreover, we assume that the global-memory drift induced by the rEMA-smoothed reduced memories and server aggregation is bounded by $2R_M/(t+1)$ under the bounded-memory condition.

Theorem 2 (Round loss reduction) *Within a fixed communication round t, under Assumptions 1–4, for any client and local iterations $e = 0,\dots,E-1$, if $\eta \in (0, 2/L_\theta)$, then*

$$\mathbb{E}\big[\mathcal{L}_E^{(t)}\big] \le \mathbb{E}\big[\mathcal{L}_0^{(t)}\big] - \Big(\eta - \tfrac{\eta^2 L_\theta}{2}\Big)\sum_{e=0}^{E-1}\mathbb{E}\Big[\|\nabla_\theta\mathcal{L}_e^{(t)}\|_2^2\Big] + \frac{E\,\eta^2 L_\theta}{2}\,\sigma^2, \tag{32}$$

and $\forall e$, $\mathbb{E}[\mathcal{L}_e^{(t)}] \le 2R_M < \infty$.

Proof 3 Assumption 4 implies $\mathrm{dis}(\mathbf{m},\mathbf{b}) \le 2R_M$, hence $\max(0,\mathrm{dis}-\tau_\ell) \le \max(0, 2R_M - \tau_\ell) \le 2R_M$, as $\tau_\ell > 0$. Therefore $\mathbb{E}[\mathcal{L}_e^{(t)}] \le 2R_M$. Let the update be $\theta_{e+1}^{(t)} = \theta_e^{(t)} - \eta\, g_e^{(t)}$. By the Assumption 1, apply the one-step smooth upper bound to $\theta \mapsto \mathbb{E}[\mathcal{L}_e^{(t)}]$:

$$\begin{aligned}\mathbb{E}[\mathcal{L}_{e+1}^{(t)}] &\le \mathbb{E}[\mathcal{L}_e^{(t)}] + \langle\nabla_\theta\mathbb{E}[\mathcal{L}_e^{(t)}], \theta_{e+1}^{(t)} - \theta_e^{(t)}\rangle \\ &\quad + \frac{L_\theta}{2}\|\theta_{e+1}^{(t)} - \theta_e^{(t)}\|_2^2 \\ &\le \mathbb{E}[\mathcal{L}_e^{(t)}] - \eta\,\mathbb{E}\Big[\langle\nabla_\theta\mathcal{L}_e^{(t)},\ g_e^{(t)}\rangle\Big] + \frac{\eta^2 L_\theta}{2}\,\mathbb{E}\|g_e^{(t)}\|_2^2.\end{aligned}$$

By Assumption 3, $\mathbb{E}[g_e^{(t)}] = \mathbb{E}[\nabla_\theta\mathcal{L}_e^{(t)}]$ and $\mathbb{E}\|g_e^{(t)}\|_2^2 = \mathbb{E}\|\nabla_\theta\mathcal{L}_e^{(t)}\|_2^2 + \sigma^2$; substituting yields

$$\mathbb{E}[\mathcal{L}_{e+1}^{(t)}] \le \mathbb{E}[\mathcal{L}_e^{(t)}] - \Big(\eta - \tfrac{\eta^2 L_\theta}{2}\Big)\mathbb{E}\|\nabla_\theta\mathcal{L}_e^{(t)}\|_2^2 + \frac{\eta^2 L_\theta}{2}\sigma^2.$$

Summing over $e = 0,\dots,E-1$ gives (32). □

Theorem 3 (FedDyMem convergence) *Under Assumptions 1–4 with fixed local steps $E \ge 1$, as $T \to \infty$, we have*

$$\frac{1}{TE}\sum_{t=1}^{T}\sum_{e=0}^{E-1}\mathbb{E}\Big[\|\nabla_\theta\mathcal{L}_e^{(t)}\|_2^2\Big] \longrightarrow 0. \tag{33}$$

Proof 4 Sum Theorem 2 over $t = 1,\dots,T$ to obtain

$$\begin{aligned}&\Big(\eta - \tfrac{\eta^2 L_\theta}{2}\Big)\sum_{t=1}^{T}\sum_{e=0}^{E-1}\mathbb{E}\|\nabla_\theta\mathcal{L}_e^{(t)}\|_2^2 \\ &\le \sum_{t=1}^{T}\big(\mathbb{E}[\mathcal{L}_0^{(t)}] - \mathbb{E}[\mathcal{L}_E^{(t)}]\big) + T\cdot\tfrac{E\,\eta^2 L_\theta}{2}\,\sigma^2.\end{aligned} \tag{34}$$

Apply a decomposition to the first term on the right:

$$\begin{aligned}\sum_{t=1}^{T}\big(\mathbb{E}[\mathcal{L}_0^{(t)}] - \mathbb{E}[\mathcal{L}_E^{(t)}]\big) &= \mathbb{E}[\mathcal{L}_0^{(1)}] - \mathbb{E}[\mathcal{L}_E^{(T)}] \\ &\quad + \sum_{t=1}^{T-1}\Big(\mathbb{E}[\mathcal{L}_0^{(t+1)}] - \mathbb{E}[\mathcal{L}_E^{(t)}]\Big),\end{aligned} \tag{35}$$

Let $\Delta_t := \mathbb{E}\|\mathbf{B}^{(t)} - \mathbf{B}^{(t-1)}\|_2$. By the 1-Lipschitz property in Assumption 4,

$$\mathbb{E}[\mathcal{L}_0^{(t+1)}] - \mathbb{E}[\mathcal{L}_E^{(t)}] \le \Delta_t. \tag{36}$$

Using the lower bound $\mathbb{E}[\mathcal{L}_E^{(T)}] \geq \underline{\ell}$, substitute back into (34) and divide both sides by $TE(\eta - \eta^2 L_\theta/2)$ to get

$$\frac{1}{TE}\sum_{t=1}^{T}\sum_{e=0}^{E-1}\mathbb{E}\Big[\|\nabla_\theta \mathcal{L}_e^{(t)}\|_2^2\Big] \leq \frac{\mathbb{E}[\mathcal{L}_0^{(1)}] - \underline{\ell}}{(\eta - \eta^2 L_\theta/2)\,TE} + \frac{\eta L_\theta}{2(\eta - \eta^2 L_\theta/2)}\,\sigma^2 + \frac{1}{(\eta - \eta^2 L_\theta/2)\,TE}\sum_{t=1}^{T-1}\Delta_t. \tag{37}$$

For the first term, with $\eta \in (0, 2/L_\theta)$ and $E \geq 1$, the numerator is constant while the denominator grows linearly in T, hence it vanishes. For the second term, either use decreasing step sizes, leading to a noise factor proportional to $(\sum_{t,e}(\eta_e^{(t)})^2)/(\sum_{t,e}\eta_e^{(t)}) \to 0$ if $\sum \eta_e^{(t)} = \infty$ and $\sum(\eta_e^{(t)})^2 < \infty$, or fix η and increase batch size so that $\sigma^2 \to 0$; in both cases the term vanishes. For the third term, using Assumption 4, we have

$$\Delta_t \leq \frac{2R_M}{t+1}.$$

Thus $\sum_{t=1}^{T-1}\Delta_t = O(\log T)$, and after normalization by TE it tends to 0. Combining the three parts proves the claim. □

The above analysis shows that the expected loss-gradient converges to zero as iterations increase, thereby proving FedDyMem–s convergence.

6 Experiments

All experiments were implemented in PyTorch and conducted on a server equipped with AMD EPYC 9554 CPUs and NVIDIA GeForce RTX A6000 GPUs.

6.1 Experimental Settings

Federated Tasks and Data Sources. We evaluate FedDyMem on six independently constructed federated UAD tasks built from eleven public industrial and medical datasets. Each task corresponds to a separate federation with a shared inspection or screening objective. The six tasks are summarized as:

- *Brain-AD Task(Brain):* We combined 3 brain image diagnostic datasets, including Brain-Tumor (Cai et al., 2024), BraTS2021 (Baid et al., 2021) and HeadCT (Salehi et al., 2021). These datasets, consisting of both MRI and CT images, were distributed across different clients in a heterogeneous manner to represent a mixture of various image types.
- *ChestXray-AD Task(X-ray):* Three diagnostic chest X-ray datasets, including ChestXRay2017 (Kermany et al., 2018), RSNA (Wang et al., 2017), and VinDr-CXR (Nguyen et al., 2022), were used to construct the ChestXray-AD task. These datasets were derived from distinct medical institutions, ensuring a diverse and heterogeneous data distribution, which is essential for the federated UAD.
- *PCB-AD Task(PCB):* We construct the PCB-AD task by combining four PCB product types from VisA (Zou et al., 2022) and two PCB product types from VISION (Bai et al., 2023). Consequently, the dataset provides a total of 6 distinct types of PCB product images, facilitating simulation in federated UAD.
- *Texture-AD Task(Texture):* This task utilizes 5 texture classes(carpet, grid, leather, tile, and wood) from the MVTec dataset (Bergmann et al., 2019), which includes mask annotations, to simulate industrial federated anomaly detection scenarios.
- *MetalParts-AD Task(MPDD)* (Jezek et al., 2021): The MPDD comprises images of 6 types of metal parts, each captured under varying conditions of spatial orientation, object positioning, and camera distance. These images are further characterized by diverse lighting intensities and non-homogeneous backgrounds, providing a comprehensive dataset for our federated UAD.
- **Wafer-AD Task(Wafer)** (Lei et al., 2025): Wafer-AD task comprises a dataset of 14 distinct wafer products, each imaged using an industrial-grade, high-resolution optical camera. This dataset also explores these wafer products under various optical conditions, thus providing a rich federated UAD simulate environment with diverse feature distributions.

Table 1 Default non-IID client partition used in the main experiments

Task	Partitioning attribute	#Groups	N	α
Brain	Data source	3	5	0.1
X-ray	Data source	3	5	0.1
PCB	PCB product type	6	5	0.1
Texture	Texture class	5	5	0.1
MPDD	Metal-part type	6	5	0.1
Wafer	Wafer product type	14	5	0.1

Client Data Partition. For each federated task, we construct non-IID client datasets from normal training samples according to task-specific source or subtype attributes. Specifically, the normal samples are first grouped by the corresponding data source, product type, texture class, metal-part type, or wafer product type, as summarized in Table 1. The main experiments use 5 simulated clients, and the samples in each group are allocated among them according to a Dirichlet distribution with $\alpha_{\text{Dir}} = 0.1$. This default partition produces

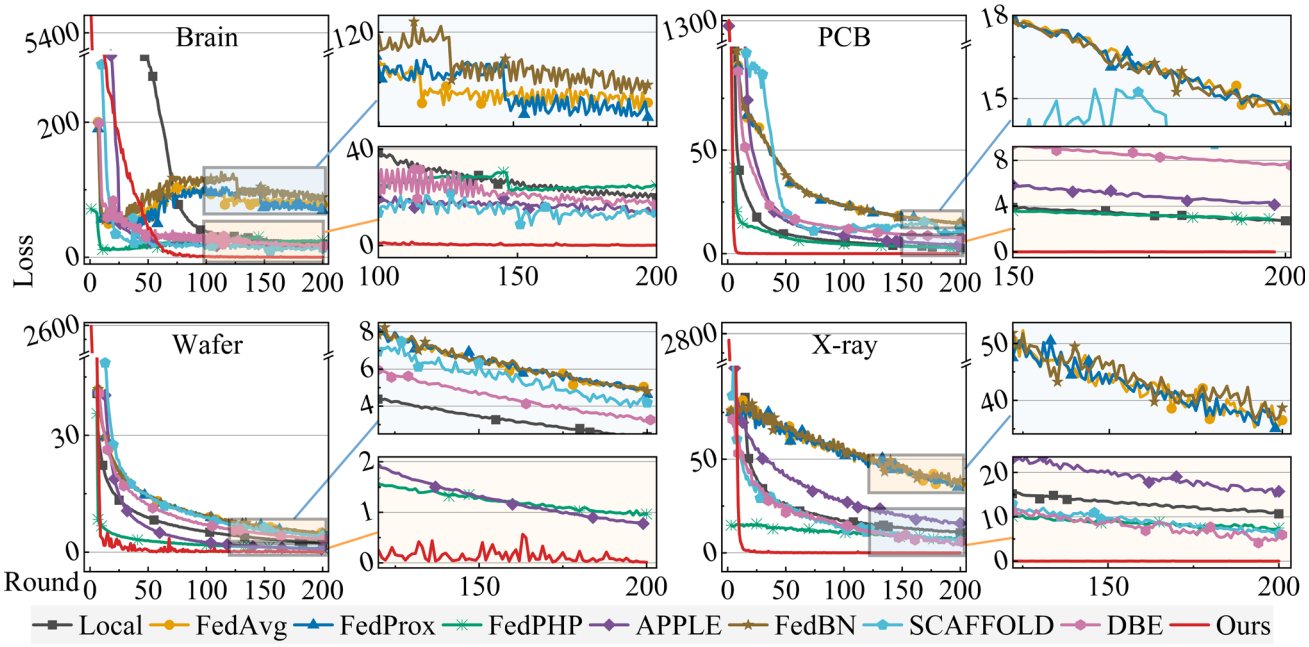

Fig. 5 Local metric loss curves of the first client over 200 communication rounds on four federated UAD tasks

non-IID client datasets in which all training samples are normal, while the proportions of different groups vary across clients. The impact of alternative client partition configurations is further examined in Sect. 6.5.

Implementation Details. We use a pretrained WideResNet-50 (Zagoruyko & Komodakis, 2016) as the frozen backbone and extract features from $L = 4$ layers. The projection layer, memory generator, and trainable grid are optimized locally, while the backbone remains fixed throughout training. The default trainable grid size is set to $H_g = W_g = 8$, the margin in the local metric loss is set to $\tau_\ell = 0.01$, and the number of nearest neighbors is set to $k_{\text{nn}} = 3$. Each uploaded reduced memory bank contains $Q = H_1 W_1$ prototypes of dimension C. The server aggregates them into $K_{\text{clu}} = Q$ global prototypes with a $I_{\text{clu}} = 15$ iterations in each aggregation round. In default setting, all five simulated clients participate in every communication round. We train for $T = 200$ communication rounds, with $E = 1$ local epoch and a batch size of 10. The trainable local modules are optimized using Adam with a learning rate of 1×10^{-3} and weight decay of 5×10^{-4}. The clients share the same initialization of model parameters. All images in the dataset were preprocessed by cropping and resizing to a resolution of 224×224 pixels using the bicubic interpolation method. All compared federated methods use the same client partitions, communication budget, local training schedule, and input preprocessing and their aggregation procedures follow their respective algorithms.

Evaluation Metric. For the evaluation of anomaly detection, we adopt the Area Under the Receiver Operating Characteristic Curve (AUROC), ensuring a comprehensive assessment of our proposed model in alignment with prior studies. Image-level anomaly detection performance is quantified using the standard AUROC metric, denoted as Image-level AUROC(I-AUROC). Additionally, to assess anomaly localization, we compute the Pixel-level AUROC(P-AUROC). To further enhance the evaluation of anomaly localization capabilities, we calculate the Per-Region Overlap (PRO) at the pixel level, providing a more granular and comprehensive analysis. These metrics are widely used in previous work (Roth et al., 2022; You et al., 2022; Cai et al., 2024).

6.2 Convergence Curves and Distribution Consistency

Figure 5 compares the training loss curves of FedDyMem and baseline methods on Brain, Wafer, PCB, and X-ray over 200 communication rounds. All methods use the same local metric loss and training configuration for fair comparison. FedDyMem converges faster and reaches lower loss values across the datasets. In the later rounds, its loss curves remain more stable with smaller fluctuations, which is consistent with the convergence analysis in Sect. 5. Figure 6 further reports I-AUROC over communication rounds on all six federated UAD tasks, showing that FedDyMem achieves higher anomaly detection performance than the baselines.

Figure 7 visualizes memory representations on the Texture task after 200 rounds. In Fig. 7a, FedAvg local memories show clear deviations from the global memory, with local losses up to 10^4 times higher than FedDyMem. This indicates that FedAvg struggles to align local memory features with

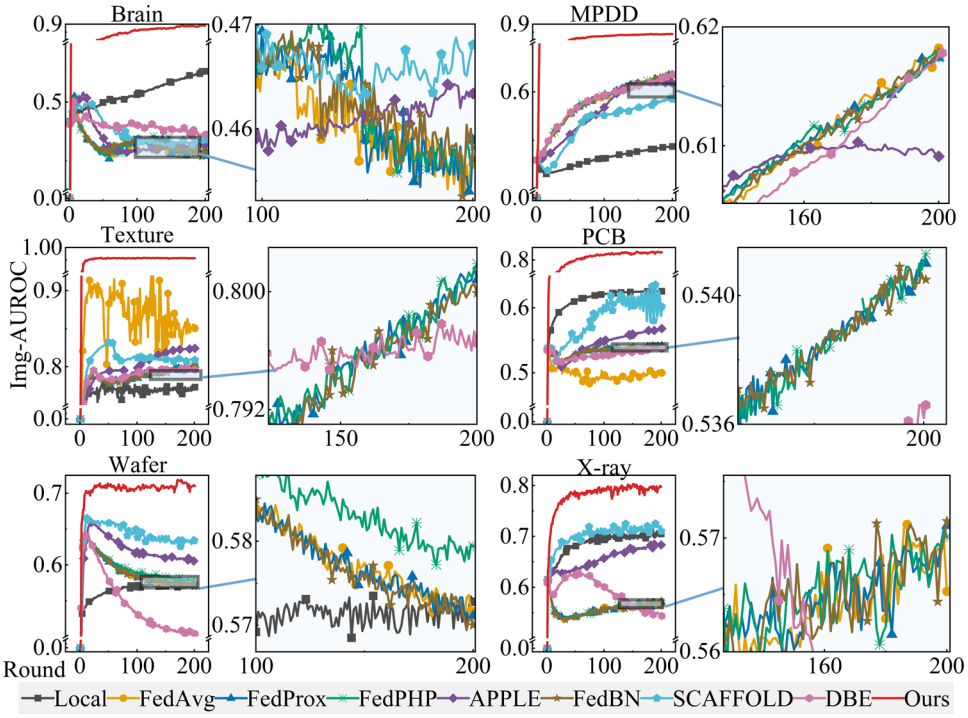

Fig. 6 Illustration of I-AUROC over communication rounds on six federated UAD tasks

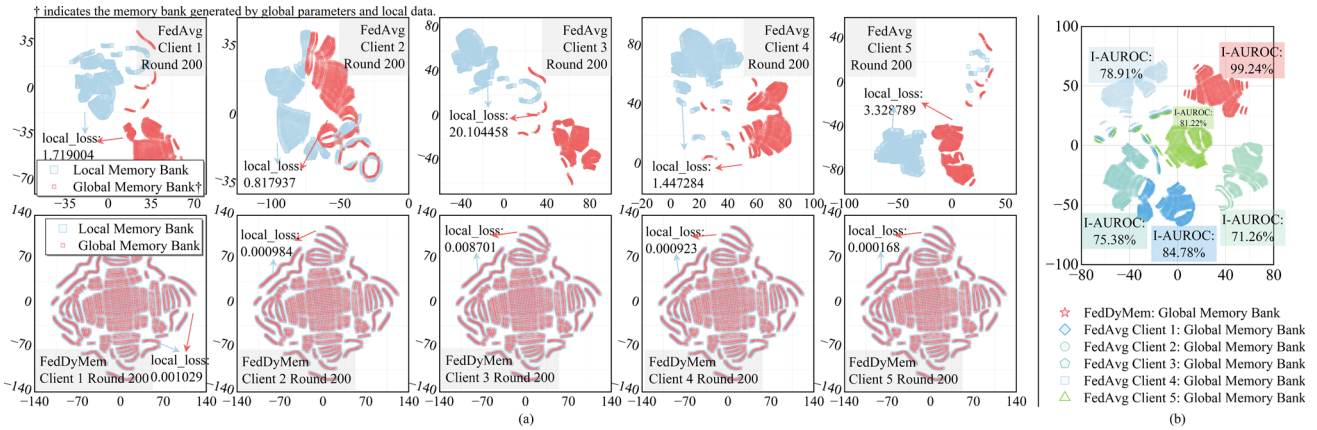

Fig. 7 Comparison of feature distributions on the texture task after 200 communication rounds. **a** t-SNE (Lvd & Hinton, 2008) visualization of local and global memory banks generated by five clients using FedAvg (top row) and FedDyMem (bottom row). **b** Comparison of global memory banks aggregated from FedDyMem and FedAvg clients, together with the corresponding I-AUROC results

the global distribution under client heterogeneity. In contrast, Fig. 7b shows that FedDyMem forms a more compact and coherent global feature space. The dynamic memory update and K-means aggregation help align client representations and reduce the effect of intra-class distribution bias.

6.3 Performance Comparisons with Existing Methods

Federated Comparison. To demonstrate the superior performance of FedDyMem, we conducted extensive comparative analyses with 8 baseline models. These include the basic federated learning algorithm FedAvg (Mcmahan et al., 2017), the regularization-based FedProx (Li et al., 2020b), the personalized-based FedPHP (Li et al., 2021b) and APPLE (Luo & Wu, 2022). Additionally, we compared against domain-skew-oriented methods such as FedBN (Li et al., 2021a) and SCAFFOLD (Karimireddy et al., 2020), as well as the model-splitting method DBE (Zhang et al., 2024a) and a purely local training scheme (Local). Table 2 reports image-level anomaly detection results. Conventional federated methods, including FedAvg, FedProx, FedPHP, and DBE, show similar I-AUROC performance with limited or inconsistent gains over Local training under heteroge-

Table 2 Comparison of Image-AUROC (%) results on six federated UAD tasks under the default non-IID protocol

Methods	Communication	Wafer	PCB	Texture	MPDD	Brain	X-ray
Local	–	57.35 (±0.71)	51.89 (±0.76)	75.94 (±0.39)	54.39 (±0.77)	50.66 (±0.87)	61.02 (±0.58)
FedAvg	Model Parameters: ≈ 10.66M, (SCAFFOLD) ≈ 10.66M × 2	63.91 (±0.36)	54.18 (±0.33)	80.32 (±0.81)	61.87 (±0.76)	50.16 (±0.63)	61.48 (±0.56)
FedProx		63.98 (±0.51)	54.11 (±0.48)	80.22 (±0.78)	61.83 (±0.61)	50.47 (±0.34)	61.45 (±0.66)
FedPHP		63.94 (±0.56)	54.13 (±0.36)	80.19 (±0.62)	61.80 (±0.63)	50.38 (±0.39)	61.45 (±0.37)
APPLE		65.92 (±0.33)	56.79 (±0.65)	82.43 (±0.51)	61.03 (±0.77)	50.82 (±0.45)	68.47 (±0.51)
FedBN		63.89 (±0.35)	54.09 (±0.79)	80.13 (±0.77)	61.90 (±0.61)	50.04 (±0.54)	61.45 (±0.72)
SCAFFOLD		66.53 (±0.85)	63.56 (±0.30)	83.26 (±0.37)	59.41 (±0.80)	49.90 (±0.52)	72.52 (±0.52)
DBE		64.12 (±0.49)	53.78 (±0.38)	79.78 (±0.90)	61.78 (±0.51)	50.06 (±0.85)	62.77 (±0.61)
Ours	Memory Size: ≈ **5.62M**	**71.96 (±0.32)**	**83.34 (±0.45)**	**99.24 (±0.36)**	**83.93 (±0.77)**	**89.27 (±0.45)**	**80.30 (±0.36)**

Bold values indicate the best results for the respective metrics

Table 3 Comparison of Pixel-AUROC (%) and PRO (%) results on four federated UAD tasks under the default non-IID protocol

Methods	Pixel-AUROC (%)				PRO (%)			
	Wafer	PCB	Texture	MPDD	Wafer	PCB	Texture	MPDD
Local	58.27 (±1.20)	68.12 (±1.11)	72.63 (±0.98)	87.02 (±1.05)	47.76 (±0.52)	45.86 (±1.31)	56.97 (±0.48)	51.31 (±0.73)
FedAvg	66.56 (±0.70)	69.30 (±0.34)	76.16 (±0.73)	91.67 (±0.94)	67.00 (±0.30)	54.04 (±0.57)	53.35 (±1.49)	55.07 (±1.25)
FedProx	66.41 (±0.74)	69.10 (±1.28)	76.08 (±1.28)	91.59 (±1.02)	67.25 (±0.33)	54.04 (±1.28)	53.62 (±0.70)	54.96 (±1.16)
FedPHP	66.44 (±1.38)	69.11 (±0.42)	76.03 (±0.57)	91.56 (±0.73)	67.16 (±0.38)	54.23 (±1.16)	53.37 (±0.47)	54.86 (±1.42)
APPLE	69.88 (±0.95)	71.34 (±0.85)	78.06 (±1.08)	92.12 (±1.19)	69.02 (±0.42)	60.11 (±1.45)	54.14 (±1.16)	55.84 (±0.76)
FedBN	66.32 (±0.57)	68.91 (±0.82)	75.92 (±1.34)	91.59 (±0.38)	67.16 (±0.68)	54.21 (±1.07)	53.48 (±0.79)	54.92 (±0.82)
SCAFFOLD	68.75 (±1.19)	77.18 (±0.54)	80.81 (±0.35)	93.47 (±0.74)	68.74 (±1.03)	66.43 (±0.58)	55.79 (±1.30)	58.71 (±0.84)
DBE	66.95 (±1.07)	72.95 (±1.35)	76.18 (±0.89)	92.66 (±1.36)	68.46 (±1.05)	57.89 (±0.75)	53.86 (±0.58)	55.76 (±0.95)
Ours	**78.40 (±1.31)**	**96.03 (±0.79)**	**97.04 (±0.56)**	**98.97 (±1.39)**	**74.91 (±0.54)**	**81.09 (±1.31)**	**86.81 (±1.13)**	**79.43 (±1.15)**

Bold values indicate the best results for the respective metrics

neous federated settings. FedDyMem achieves the highest I-AUROC across all six tasks, reaching 71.96% on Wafer, 83.34% on PCB, 99.24% on Texture, 83.93% on MPDD, 89.27% on Brain, and 80.30% on X-ray. It also reduces communication cost by exchanging a reduced memory bank with about 5.62M elements, compared with about 10.66M parameters in conventional parameter-sharing methods. Under the same numerical precision, this gives a 47.3% reduction in the nominal size of the primary exchanged representation.

Table 3 reports anomaly localization results on datasets with pixel-level annotations. FedDyMem achieves the highest mean P-AUROC and PRO on Wafer, PCB, Texture, and MPDD. Specifically, it obtains P-AUROC scores of 78.40%, 96.03%, 97.04%, and 98.97%, with PRO scores of 74.91%, 81.09%, 86.81%, and 79.43%, respectively.

Representative UAD Comparison. We further compare FedDyMem with representative UAD methods, including PatchCore (Roth et al., 2022), SimpleNet (Liu et al., 2023), UniAD (You et al., 2022), MambaAD (He et al., 2024), and InpFormer (Luo et al., 2025), together with reference variants of the FedDyMem architecture, in both centralized and federated settings. In the centralized setting, all normal training samples from the participating clients are jointly available for training. For the federated setting, SimpleNet, UniAD, MambaAD, and InpFormer are optimized using FedAvg under the same client partition and communication schedule. PatchCore uses a frozen feature extractor and a non-parametric memory bank, and thus does not support iterative parameter aggregation through FedAvg. Therefore, each client extracts patch features once from its local normal samples, and the server aggregates the local patch features to construct a global memory bank for anomaly detection. Table 4 reports the comparison results. The two Ours† variants use the same modules as the local clients in FedDyMem. In the centralized setting, PatchCore provides a patch feature reference, SimpleNet represents a single-model UAD method, and UniAD, MambaAD, and InpFormer represent unified UAD models. For our centralized variant, all normal training samples are

Table 4 Comparison with representative UAD methods under centralized and federated settings

Methods	Setting	Wafer	PCB	Texture	MPDD	Brain	X-ray
All normal training data are jointly accessible.							
PatchCore	Centralized	75.56/77.92/71.80	91.25/96.12/77.97	99.04/97.46/88.76	85.00/98.88/71.31	95.69/–/–	78.10/–/–
SimpleNet	Centralized	69.92/72.16/70.51	84.35/96.17/70.47	99.32/96.43/85.59	86.91/98.92/74.65	92.79/–/–	77.96/–/–
UniAD	Centralized	74.05/83.23/76.56	83.79/96.74/79.59	98.92/94.90/89.63	69.28/87.65/70.06	87.51/–/–	**84.89**/–/–
MambaAD	Centralized	76.34/84.69/84.67	94.59/**99.11/91.52**	98.41/97.99/88.15	88.11/96.74/90.58	97.73/–/–	82.67/–/–
InpFormer	Centralized	**81.85/85.02/85.05**	**94.74**/98.87/88.36	**99.82/98.67/95.56**	**91.09/99.55/94.56**	**98.73**/–/–	83.44/–/–
Ours†	Centralized	73.42/79.12/75.67	84.45/96.72/82.12	99.02/97.66/87.21	85.73/95.29/80.14	90.36/–/–	81.11/–/–
Raw normal training data remain client-local.							
PatchCore	–	68.17/69.03/68.34	71.59/91.06/65.47	90.31/86.01/82.53	69.55/84.41/59.76	71.16/–/–	68.22/–/–
SimpleNet	FedAvg	64.28/63.64/67.68	65.03/83.13/62.90	88.20/85.79/71.63	69.02/88.75/63.70	69.27/–/–	67.58/–/–
UniAD	FedAvg	65.33/68.02/67.57	62.25/90.08/64.04	88.39/84.97/80.34	63.18/85.85/56.44	73.93/–/–	68.70/–/–
MambaAD	FedAvg	69.77/72.33/70.67	73.47/91.13/73.25	93.48/92.88/81.63	77.12/90.47/66.38	86.82/–/–	75.85/–/–
InpFormer	FedAvg	70.98/74.05/72.90	78.81/93.26/77.78	97.48/92.88/83.63	80.20/94.36/75.13	88.90/–/–	78.90/–/–
Ours†	FedAvg	63.91/66.56/67.00	54.18/69.30/54.04	80.32/76.16/53.35	61.87/91.67/55.07	50.16/–/–	61.48/–/–
Ours	FedDyMem	**71.96/78.40/74.91**	**83.34/96.03/81.09**	**99.24/97.04/86.81**	**83.93/98.97/79.43**	**89.27**/–/–	**80.30**/–/–

Results are reported as the mean I-AUROC/P-AUROC/PRO (%). †Reference variants of the FedDyMem architecture, which use the same underlying trainable modules as FedDyMem
Bold values indicate the highest scores within the centralized and federated sections separately

pooled and optimized jointly without client partitioning or communication. The memory is updated iteratively within this centralized training process, and the memory from the previous memory-update iteration is used as the reference. Under joint data access, MambaAD and InpFormer obtain strong centralized performance, confirming the effectiveness of unified UAD models when heterogeneous normal data are jointly observable during optimization. However, their performance is substantially reduced when moving from centralized joint optimization to the federated FedAvg setting. Averaged over the six tasks, UniAD, MambaAD, and InpFormer lose 12.78, 10.22, and 9.07 I-AUROC points, respectively. This systematic gap indicates that unified modeling capacity alone is insufficient to reproduce centralized joint optimization in the federated setting, where each local update is optimized on a client-specific normal distribution and parameter averaging is not explicitly designed to preserve coverage of complementary normal modes across clients. The same-architecture comparison further supports this observation. Our centralized architecture loses 23.70 I-AUROC points on average under FedAvg, consistent with the difficulty of averaging locally optimized modules across heterogeneous client distributions. Although clients start from the same initialization, non-IID local updates can specialize the projection, memory-generator, and grid modules to different normal modes. Averaging these nonlinear modules does not necessarily preserve the normal representation geometry learned by individual clients, which can result in incomplete federation-wide normal-mode coverage. Unlike this Ours†+FedAvg ablation, the decentralized PatchCore reference uses a frozen shared feature extractor and aggregates local patch representations in a common pretrained feature space, thereby avoiding the averaging of client-specialized trainable modules. By keeping these modules client-specific and sharing reduced normal memories through server-side aggregation, FedDyMem narrows the centralized-to-federated gap to only 1.01 points on average while achieving the best federated results across all six image-level tasks and four localization tasks.

Table 5 Comparison under the group-disjoint federated setting

Methods	Setting	Wafer ($N = 14$)	Brain ($N = 3$)
UniAD	FedAvg	67.48/76.06/61.75	74.81/–/–
MambaAD	FedAvg	65.91/69.32/55.66	86.42/–/–
InpFormer	FedAvg	64.15/71.16/53.45	83.31/–/–
Ours	FedDyMem	**70.20/77.62/73.37**	**88.24**/–/–

Results are reported as the mean I-AUROC/P-AUROC/PRO (%)
Bold values indicate the best results for the respective metrics

Group-Disjoint Federated Comparison. To further examine federated UAD when normal modes are not jointly observable during local training, we construct a stricter group-disjoint setting on Wafer and Brain. Specifically, each wafer product type is assigned exclusively to one client, resulting in $N = 14$, while each of the three data sources in Brain is assigned to one client, resulting in $N = 3$. Thus, each client observes only one task-specific group during training, while the federation collectively covers all groups. Unlike

Table 6 Impact of per-round local training sample ratio

Ratio	Methods	Brain	MPDD	Texture
20%	FedAvg	50.06/–/–	58.65/90.72/54.22	77.18/72.64/50.53
	Ours	85.05/–/–	79.60/97.97/75.68	98.34/96.92/81.82
40%	FedAvg	50.28/–/–	58.27/90.41/54.11	78.50/74.16/50.20
	Ours	86.49/–/–	81.23/97.97/77.54	99.09/96.89/82.53
60%	FedAvg	50.08/–/–	60.62/91.01/54.20	78.44/74.14/51.75
	Ours	87.59/–/–	82.96/97.94/78.91	99.09/96.86/83.47
80%	FedAvg	50.16/–/–	61.01/91.16/54.95	80.28/75.33/52.01
	Ours	88.47/–/–	83.19/98.83/79.23	99.21/96.51/85.82
100%	FedAvg	50.16/–/–	61.87/91.67/55.07	80.32/76.16/53.35
	Ours	89.27/–/–	83.93/98.97/79.43	99.24/97.04/86.81

Results are reported as the mean I-AUROC/P-AUROC/PRO (%)

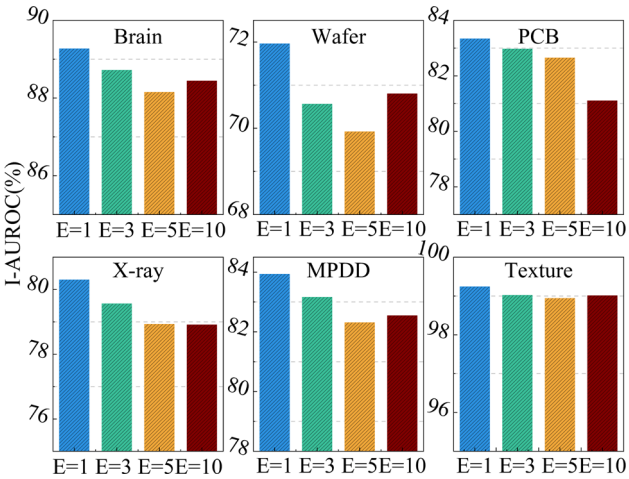

Fig. 8 Impact of different local epochs on clients

centralized unified UAD, where normal samples from different groups can jointly participate in optimization, local updates in this setting cannot directly access normal modes held by other clients.

Table 5 compares FedDyMem with representative unified UAD methods adapted using FedAvg. FedDyMem achieves the best I-AUROC on both tasks, improving over the strongest FedAvg baseline by 2.72 points on Wafer and 1.82 points on Brain. On Wafer, it also improves PRO from 61.75% to 73.37%. Together with the centralized-to-federated comparison in Table 4, these results provide further empirical support that the challenge considered in federated UAD arises not merely from modeling multiple normal groups, but from integrating complementary normal modes that are distributed across clients and are not jointly observable during local optimization.

6.4 Ablation Study

Impact of Local Training. Figure 8 illustrates the impact of varying the number of local training epochs (E) on the I-AUROC across all datasets. Across the evaluated tasks, $E = 1$ achieves the highest I-AUROC. For example, in the Wafer dataset, accuracy drops from 71.96% at $E = 1$ to 69.92% at $E = 5$. Similarly, the PCB dataset demonstrates a decrease in performance, with accuracy falling from 83.34% at $E = 1$ to 82.65% at $E = 5$. MPDD, Brain, and X-ray also obtain lower I-AUROC values when using more than one local epoch. This phenomenon can be attributed to the heterogeneous data distribution across clients. As the number of local training epochs increases, the local models tend to overfit their respective client datasets, leading to a decrease in the effectiveness of the globally aggregated memory features.

Table 6 further evaluates whether FedDyMem remains effective when only a fraction of local normal samples is used in each communication round. With $E = 1$, FedDyMem consistently outperforms FedAvg at all evaluated sample ratios. For example, its I-AUROC increases from 85.05% to 88.47% on Brain, from 79.60% to 83.19% on MPDD, and from 98.34% to 99.21% on Texture when the ratio increases from 20 to 80%. Therefore, using a reduced subset of local samples can retain strong performance, whereas increasing the per-round sample coverage yields better I-AUROC in the evaluated range.

Impact of rEMA and Dynamic Weighting in Memory-reduce. Table 7 presents the ablation study of rEMA and Dynamic Weighting (DW) in the memory-reduce module. All other FedDyMem network components and the server-side aggregation method are retained in all variants. When DW is removed, the local memory features are reduced by uniform averaging. Both rEMA and DW improve the performance over the variant without either component. For example, on Brain, using rEMA alone and DW alone improves the I-AUROC from 84.47% to 86.75% and 87.62%, respectively.

Table 7 Ablation study of rEMA and Dynamic Weighting (DW) in the memory-reduce

rEMA	DW	Brain	MPDD	Texture
-	-	84.47/–/–	78.63/95.72/74.17	97.91/95.32/83.13
✓	–	86.75/–/–	80.48/98.88/74.85	98.66/96.88/85.67
-	✓	87.62/–/–	81.88/97.85/77.73	98.99/96.99/85.88
✓	✓	**89.27**/–/–	**83.93/98.97/79.43**	**99.24/97.04/86.81**

Bold values indicate the best results for the respective metrics

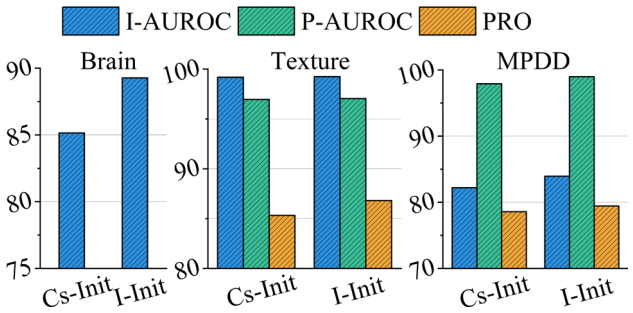

Fig. 9 Impact of parameter initialization

Table 8 Impact of hyperparameters and backbones

Factor	Setting	PCB	Brain
$H_g \times W_g$	2×2	79.02/93.38/78.89	86.27/–/–
	4×4	82.24/95.74/80.96	88.73/–/–
	8×8	83.34/**96.03/81.09**	**89.27**/–/–
	16×16	**83.36**/95.87/80.99	88.93/–/–
k_{nn}	1	78.84/93.99/78.31	86.15/–/–
	3	**83.34/96.03/81.09**	**89.27**/–/–
	5	83.20/95.98/80.55	89.23/–/–
	7	83.28/95.57/80.93	89.06/–/–
τ_ℓ	0.001	77.21/93.95/78.08	86.73/–/–
	0.01	**83.34/96.03/81.09**	**89.27**/–/–
	0.02	82.42/95.41/80.87	88.54/–/–
	0.05	80.17/94.25/78.99	87.68/–/–
α_t	1	82.39/94.93/79.38	87.62/–/–
	$1/(t+1)$	**83.34/96.03/81.09**	**89.27**/–/–
	$1/\sqrt{t+1}$	83.05/95.26/80.34	89.15/–/–
Backbone	ResNet-18	80.82/94.98/79.05	87.56/–/–
	ResNet-50	83.02/95.98/81.05	88.28/–/–
	EfficientNet-B4	83.39/95.43/80.38	89.21/–/–
	EfficientNet-B5	**83.51**/95.41/80.86	88.92/–/–
	WideResNet-50	83.34/**96.03/81.09**	**89.27**/–/–

Results are reported as I-AUROC/P-AUROC/PRO (%)
Bold values indicate the highest scores within each hyperparameter

Similar improvements are observed on MPDD and Texture. Combining rEMA and DW achieves the best performance across all datasets, reaching I-AUROC values of 89.27%, 83.93%, and 99.24% on Brain, MPDD, and Texture, respectively. These results indicate that rEMA and DW are both beneficial for constructing the reduced memory bank.

Table 9 Effectiveness and scalability of K-means

Configuration	N	$\bar{T}_{\text{agg}}$	Texture
Average Aggregation	5	0.0241	92.76/96.10/84.42
$K_{\text{clu}} = Q,\ I_{\text{clu}} = 5$	5	0.2373	97.94/95.77/85.12
$K_{\text{clu}} = Q,\ I_{\text{clu}} = 10$	5	0.2885	99.09/97.07/86.01
$K_{\text{clu}} = Q,\ I_{\text{clu}} = 15$	5	0.3429	99.24/97.04/**86.81**
$K_{\text{clu}} = Q,\ I_{\text{clu}} = 20$	5	0.3987	99.22/97.05/**86.81**
$K_{\text{clu}} = Q,\ I_{\text{clu}} = 40$	5	0.5375	99.27/97.03/86.77
$K_{\text{clu}} = Q/2,\ I_{\text{clu}} = 15$	5	0.2377	98.94/96.77/86.12
$K_{\text{clu}} = 2Q,\ I_{\text{clu}} = 15$	5	**0.5199**	**99.28**/96.96/86.45
Average Aggregation	3	**0.0096**	93.20/95.87/84.62
$K_{\text{clu}} = Q,\ I_{\text{clu}} = 15$	3	0.1675	99.13/**97.62**/85.32
Average Aggregation	10	0.0705	93.57/95.84/84.52
$K_{\text{clu}} = Q,\ I_{\text{clu}} = 15$	10	0.4972	99.16/96.89/85.88

Results are reported as the mean per-round server aggregation time $\bar{T}_{\text{agg}}$ (s) measured over 200 communication rounds and I-AUROC/P-AUROC/PRO (%)
Bold values indicate the lowest aggregation time and the highest scores across the whole table

Impact of Parameter Initialization. Figure 9 compares client-specific initialization (Cs-init), where different random seeds are used across clients, with identical initialization (I-init), where all clients share the same initialization. I-init consistently achieves better performance on all evaluated datasets. For example, the I-AUROC on Brain improves from 85.15% to 89.27%. On Texture, I-init improves the I-AUROC/P-AUROC/PRO from 99.18%/96.96%/85.32% to 99.24%/97.04%/86.81%, and on MPDD from 82.20%/97.90%/78.56% to 83.93%/98.97%/79.43%. These results indicate that identical initialization provides a more stable starting point for constructing the initial global memory.

Impact of Hyperparameters and Backbones. Table 8 evaluates the sensitivity of FedDyMem to key hyperparameters and backbone choices. Results are reported as the mean I-AUROC/P-AUROC/PRO (%). Overall, the default settings provide the most balanced performance on PCB and Brain. Increasing the grid size from 2×2 to 8×8 improves I-AUROC from 79.02 to 83.34% on PCB and from 86.27% to 89.27%

Table 10 Ablation study of structure for FedDyMem

+Proj	+MG	+KM	I-AUROC (%)							P-AUROC (%)		
			Wafer	PCB	Texture	MPDD	Brain	X-ray	Wafer	PCB	Texture	MPDD
–	–	–	67.69	69.58	85.86	70.41	69.03	68.92	70.23	83.23	85.22	94.19
✓	–	–	68.93	74.10	87.27	78.28	74.95	72.71	74.29	89.26	90.84	95.77
✓	✓	–	70.96	81.26	92.76	81.05	85.94	78.12	77.01	94.97	96.10	97.96
✓	✓	✓	**71.96**	**83.34**	**99.24**	**83.93**	**89.27**	**80.30**	**78.40**	**96.03**	**97.04**	**98.97**

Bold values indicate the best results for the respective metrics

on Brain. Although 16×16 yields a marginally higher I-AUROC on PCB, 8×8 achieves better overall localization performance and the best Brain result. Similarly, $k_{\text{nn}} = 3$, $\tau_\ell = 0.01$, and $\alpha_t = 1/(t+1)$ achieve the best or most balanced results among the evaluated settings. For backbone choices, FedDyMem remains effective with ResNet (He et al., 2016), EfficientNet (Tan & Le, 2019), and WideResNet backbones. WideResNet-50 achieves the best overall performance, with the highest Brain I-AUROC (89.27%) and the best PCB P-AUROC/PRO (96.03%/81.09%). Although EfficientNet-B5 obtains a slightly higher PCB I-AUROC (83.51%), WideResNet-50 provides the most balanced performance and is adopted as the default backbone.

Effectiveness and Scalability of K-means. Table 9 evaluates the effectiveness and server-side cost of K-means aggregation on Texture. Each client uploads $Q = H_1 W_1$ reduced patch-level prototypes. Average Aggregation produces Q global prototypes by averaging the N local memories, whereas K-means clusters the received NQ prototypes into K_{clu} global prototypes. Thus, K_{clu} is set relative to Q, and $K_{\text{clu}} = Q$ keeps the same global memory capacity as Average Aggregation. With $N = 5$, $K_{\text{clu}} = Q$, and $I_{\text{clu}} = 15$, K-means improves I-AUROC/P-AUROC/PRO from 92.76%/96.10%/84.42% to 99.24%/97.04%/86.81%. Increasing I_{clu} beyond 15 brings only marginal changes but increases aggregation time. Using $K_{\text{clu}} = Q/2$ is faster but degrades performance, while $K_{\text{clu}} = 2Q$ increases cost without consistent gains in P-AUROC or PRO. Therefore, $K_{\text{clu}} = Q$ and $I_{\text{clu}} = 15$ provide a balanced trade-off. The server clusters only NQ reduced prototypes per round with complexity $O(I_{\text{clu}} NQ K_{\text{clu}} C)$. Under the default setting, the mean aggregation time increases from 0.1675 s at $N = 3$ to 0.4972 s at $N = 10$, remaining below 0.5 s per round in the evaluated range. Compared with Average Aggregation, K-means adds server-side computation but consistently improves anomaly detection performance.

Ablation Study of FedDyMem Components. Table 10 evaluates the effects of the projection layer (Proj), memory generator (MG), and server-side K-means aggregation (KM). The baseline employs memory-reduce with average aggregation, without these additional components. The results show that progressively introducing Proj, MG, and KM consistently improves anomaly detection and localization performance.

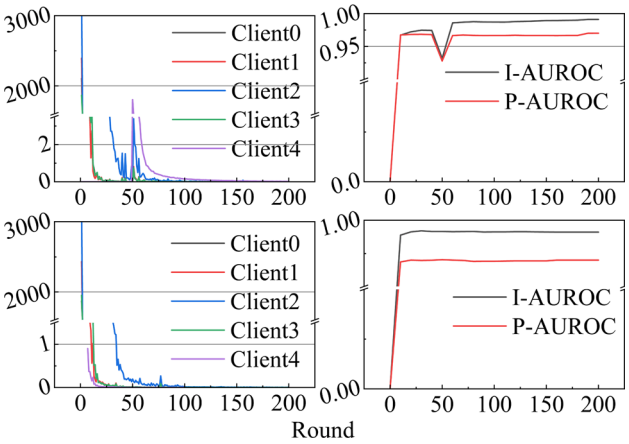

Fig. 10 Robustness to dynamic client participation

For example, on Wafer, the I-AUROC/P-AUROC increases from 67.69%/70.23% for the baseline to 68.93%/74.29% after adding Proj, and further to 70.96%/77.01% after incorporating MG. Adding KM finally improves these metrics to 71.96%/78.40%. Similar improvements are observed on PCB, where the I-AUROC increases from 74.10% with Proj to 81.26% after adding MG and 83.34% after further applying KM. Overall, the three components provide complementary improvements to the FedDyMem framework.

6.5 Robustness under Federated Deployment

Robustness to Contaminated Local Training Data. Table 11 evaluates robustness by injecting anomalous test samples into each client's normal training set. FedDyMem remains stable and consistently outperforms FedAvg across the evaluated contamination ratios. When contamination increases from 0% to 10%, its I-AUROC only decreases from 89.27% to 88.01% on Brain, 83.93% to 82.94% on MPDD, and 99.24% to 97.98% on Texture. Localization is also preserved, with PRO decreasing by only 0.44% on MPDD and 1.76% on Texture. This stability may benefit from memory-reduce and server-side K-means, which summarize local features and dilute sparse client-specific contaminated features.

Robustness to Client Participation and Data Heterogeneity. Figure 10 evaluates dynamic client participation during fed-

Table 11 Robustness to contaminated local training data

Ratio	Methods	Brain	MPDD	Texture
0%	FedAvg	50.16/–/–	61.87/91.67/55.07	80.32/76.16/53.35
	Ours	89.27/–/–	83.93/98.97/79.43	99.24/97.04/86.81
1%	FedAvg	50.14/–/–	59.43/89.91/54.91	77.97/73.93/49.74
	Ours	88.90/–/–	83.86/98.76/79.40	98.97/96.95/85.85
5%	FedAvg	50.12/–/–	58.49/88.07/53.24	72.96/68.34/47.59
	Ours	88.50/–/–	83.52/98.47/79.55	98.35/96.87/85.44
10%	FedAvg	50.04/–/–	56.73/85.46/46.94	70.97/65.00/49.46
	Ours	88.01/–/–	82.94/97.94/78.99	97.98/97.19/85.05

Ratio denotes the proportion of contaminated samples included in each client's local training set. Results are reported as the mean I-AUROC/P-AUROC/PRO (%)

Table 12 Local adaptation for a newly arriving client after federated training

Dataset	Local-only (w/o $\mathbf{B}^{(T)}$)	Local Adaptation(w/ $\mathbf{B}^{(T)}$)
Wafer	57.35/58.27/47.76	71.66/75.96/73.11
PCB	51.89/68.12/45.86	82.88/95.67/79.96
Texture	75.94/72.63/56.97	99.11/96.80/85.91
MPDD	54.39/87.02/51.31	81.83/98.92/78.36
Brain	50.66/–/–	87.77/–/–
X-ray	61.02/–/–	79.71/–/–

Results are reported as the mean I-AUROC/P-AUROC/PRO (%)

Table 13 Performance under different client counts N and Dirichlet partition parameters α_{Dir}

Method	N	α_{Dir}	Texture	MPDD
FedAvg	5	0.01	61.22/56.09/52.94	55.36/86.95/50.45
Ours	5	0.01	99.12/96.85/85.30	83.00/98.90/79.15
FedAvg	5	0.5	89.42/79.24/63.50	63.07/91.31/67.34
Ours	5	0.5	99.14/96.85/85.48	84.80/98.99/79.59
FedAvg	5	1.0	97.26/86.67/77.35	63.81/92.37/70.89
Ours	5	1.0	99.13/96.87/87.07	84.58/98.96/79.81
FedAvg	3	0.1	86.72/72.02/60.83	68.47/94.77/67.87
Ours	3	0.1	99.13/97.62/85.32	83.24/98.91/77.77
FedAvg	5	0.1	80.32/76.16/53.35	61.87/91.67/55.07
Ours	5	0.1	99.24/97.04/86.81	83.93/98.97/79.43
FedAvg	10	0.1	77.94/75.18/58.13	60.20/94.78/54.74
Ours	10	0.1	99.16/96.89/85.88	83.30/98.99/79.08

erated training. When Client-4 joins at round 50, the evaluation curves show a short fluctuation and then stabilize, reaching final I-AUROC/P-AUROC values of 99.11%/97.01%. When Client-4 drops out at round 80, the final performance remains stable at 99.12%/97.00%. Table 12 evaluates a new client arriving after federated training. Using the final global memory $\mathbf{B}^{(T)}$ as the reference memory for local adaptation consistently improves performance over local-only training. For example, PCB improves from 51.89%/68.12%/45.86% to 82.88%/95.67%/79.96% in I-AUROC/P-AUROC/PRO, and Brain I-AUROC improves from 50.66% to 87.77%. These results show that the global memory provides useful shared knowledge for newly arriving clients after local adaptation. Table 13 evaluates different Dirichlet parameters α_{Dir} and client numbers N. With $N = 5$ and α_{Dir} varying from 0.01 to 1.0, FedDyMem maintains I-AUROC within 99.12%–99.14% on Texture and 83.00%–84.80% on MPDD. FedAvg is more sensitive to severe non-IID partitions, with Texture I-AUROC dropping from 97.26% at $\alpha_{\text{Dir}} = 1.0$ to 61.22% at $\alpha_{\text{Dir}} = 0.01$. When N changes from 3 to 10, FedDyMem remains stable, with I-AUROC ranging from 99.13% to 99.24% on Texture and from 83.24% to 83.93% on MPDD. These results support robustness to the evaluated client participation changes, data heterogeneity, and federation scales.

Robustness to Rare Normal Modes. To examine whether server-side aggregation suppresses rare normal modes, we construct a rare-type Texture setting by removing the *carpet* subtype, repartitioning the remaining data among five clients with $\alpha = 0.1$, and injecting a fraction ρ of *carpet* training samples into one client. Table 14 reports performance on the rare *carpet* subtype and the corresponding Texture average. With only $\rho = 5\%$ (14 samples), FedDyMem achieves 96.11%/93.50%/80.61% I-AUROC/P-AUROC/PRO on *carpet*. When ρ increases to 100%, the *carpet* performance improves to 99.10%/98.42%/86.71%, and the Texture average increases from 97.86%/94.75%/81.42% to 99.03%/96.96%/86.73%. These results indicate that, under the evaluated setting, rare normal subtypes can remain represented in the shared memory rather than being absorbed by dominant normal modes.

Table 14 Robustness to a rare normal mode

Injected *carpet* ratio ρ (# samples)	*Carpet*	Average
5% (14)	96.11/93.50/80.61	97.86/94.75/81.42
10% (28)	96.91/93.76/80.80	97.87/94.99/81.23
20% (56)	97.47/96.15/81.54	98.24/95.10/83.01
50% (140)	98.79/97.42/86.01	98.52/95.28/84.53
100% (279)	**99.10/98.42/86.71**	**99.03/96.96/86.73**

Results are reported as I-AUROC/P-AUROC/PRO (%)
Bold values indicate the best results for the respective metrics

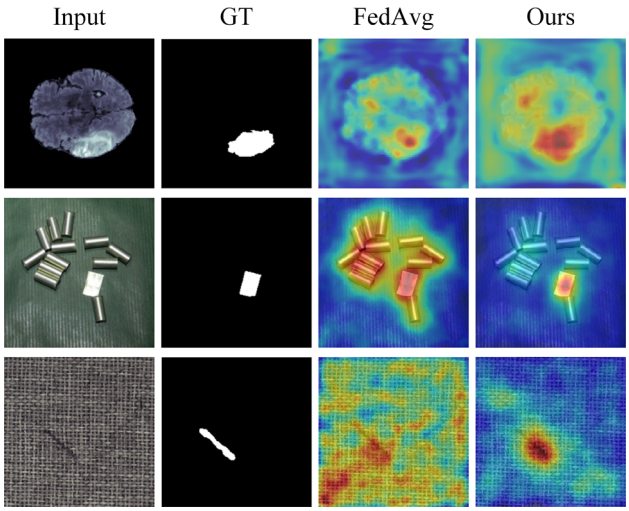

Fig. 11 Qualitative comparison of FedAvg and our proposed method. The results were obtained from a randomly selected client using a global memory bank

6.6 Qualitative Results

Figure 11 provides a qualitative comparison of the prediction results between FedAvg and the proposed FedDyMem method under an identical experimental setting. The FedDyMem approach exhibits a notable improvement in prediction quality, with substantially fewer noisy regions evident in the results. Following the methodology in Lee et al. (2022), the anomaly score heatmaps are upsampled to match the spatial resolution of the input image and subsequently refined using Gaussian filtering with $\sigma = 4$, ensuring smoother and more coherent boundaries. Furthermore, min-max normalization is applied to standardize the range of anomaly scores, facilitating clearer visualization and interpretation of the results.

6.7 Privacy Verification

Threat Model and Privacy Scope. We consider a passive honest-but-curious server that follows the protocol, records communicated messages, and performs offline inference after training. At round t, the server observes

$$\mathcal{V}_{\text{srv}} = \left\{ \left(\{\mathbf{R}_n^{(t)}\}_{n \in \mathcal{C}_t} \right) \right\}_{t=0}^{T}, \tag{38}$$

where $\mathcal{C}_t$ denotes the participating clients and $\mathbf{R}_n^{(t)}$ is the uploaded memory of client n. The attacker knows the public protocol, frozen backbone, architecture, initialization, optimization settings, and aggregation rule. It may access

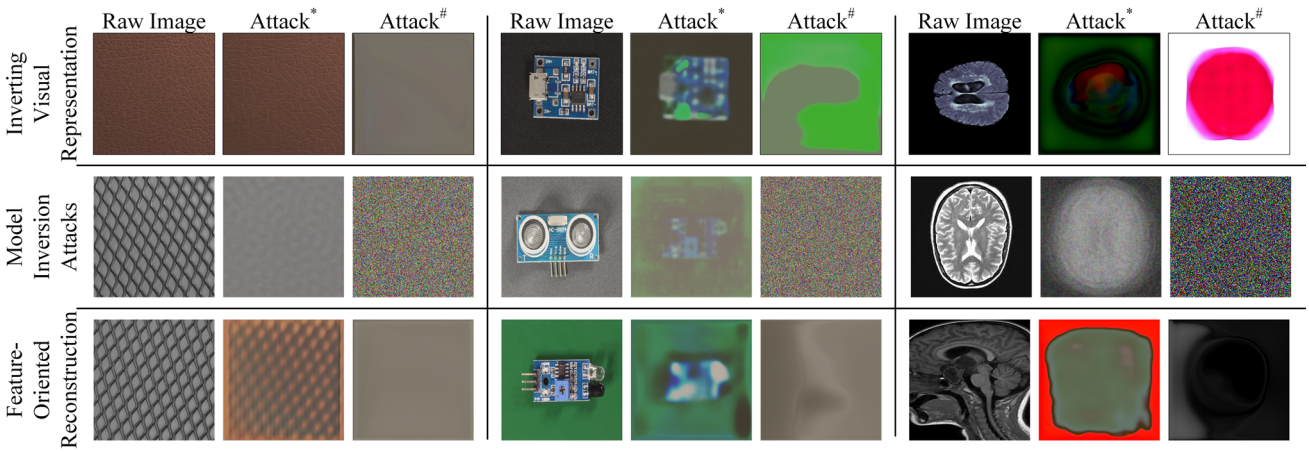

Fig. 12 Qualitative reconstruction results before and after applying the proposed memory-reduce operation on Texture, PCB, and Brain

Table 15 Quantitative reconstruction attack results before and after memory-reduce

Attack Methods	Dataset	+Memory-Reduce	SSIM↓	PSNR↓	FID↑
Deconvolution Inverting Method	Texture	–	0.6350	25.9003	246.9916
		✓	0.2088	17.4787	415.9250
	PCB	-	0.5213	25.8625	281.5855
		✓	0.3379	12.1827	358.7053
	Brain	-	0.2273	11.0985	300.5543
		✓	0.0872	2.8752	338.1136
Collaborative Inference Attack	Texture	–	0.0928	15.5441	300.9797
		✓	0.0179	11.4840	350.8446
	PCB	-	0.4860	13.9388	313.1274
		✓	0.0178	10.7200	418.3834
	Brain	–	0.1218	11.2046	365.9037
		✓	0.0038	6.5987	396.2980
Feature Reconstruction Attack	Texture	-	0.2094	16.2157	231.2310
		✓	0.0869	15.3695	326.0596
	PCB	–	0.5391	17.4453	327.5368
		✓	0.2073	14.2317	370.4590
	Brain	–	0.5629	20.2312	240.1137
		✓	0.1558	7.8485	352.5879

Table 16 Membership inference on the texture task under the passive honest-but-curious server threat model

Method	Server-visible Message	ROC-AUC ↓	TPR@1%FPR ↓	Advantage ↓
FedAvg	$\Delta\boldsymbol{\theta}_n^{(T)}$	0.6024 ± 0.0206	0.0619 ± 0.0104	0.1687 ± 0.0420
Raw-memory Sharing	$\{\mathbf{M}_{n,i}^{(T)}\}_i$	0.7537 ± 0.0403	0.3936 ± 0.0125	0.6583 ± 0.0428
FedDyMem	$\mathbf{R}_n^{(T)}$	$\mathbf{0.5310} \pm 0.0223$	$\mathbf{0.0288} \pm 0.0122$	$\mathbf{0.0960} \pm 0.0390$

Values are reported as mean ± 95% normal-approximation confidence intervals over five held-out federations

candidate images and auxiliary data from the same broad domain, but not the target client's raw training images or client-resident trainable parameters.

Feature Inversion Attacks. We evaluate visual-content leakage by reconstructing images from server-visible memory representations using three inversion attacks, including deconvolution-based inversion Dosovitskiy and Brox (2016), collaborative inference attack He et al. (2019), and stealthy feature-oriented reconstruction attack Xu et al. (2024). Reconstruction quality is measured by SSIM, PSNR, and FID, where lower SSIM and PSNR or higher FID indicate lower visual leakage. Table 15 shows that memory-reduce consistently lowers SSIM and PSNR and increases FID across the evaluated datasets and attacks. Figure 12 also shows that reconstructions from reduced memories contain less recognizable structure and texture. These results indicate that memory-reduce reduces empirical visual-content leakage under the evaluated inversion attacks.

Membership Inference Attacks. We next evaluate whether the server can infer whether a candidate normal image belongs to a target client's local training set:

$$\mathcal{A}_{\text{mem}}\left(x, \mathcal{V}_n^{(T)}\right) \to \hat{y}_{\text{mem}}, \quad y_{\text{mem}} = \mathbb{I}\left[x \in \mathcal{D}_n^{\text{train}}\right]. \tag{39}$$

We compare three server-visible messages on the texture task, including FedAvg parameter updates $\Delta\boldsymbol{\theta}_n^{(T)}$, unreduced raw memories $\{\mathbf{M}_{n,i}^{(T)}\}_i$, and FedDyMem reduced memories $\mathbf{R}_n^{(T)}$. All settings use the same Deep Sets attack classifier Zaheer et al. (2017), with leave-one-federation-out evaluation over five independently repartitioned shadow federations. As shown in Table 16, raw memory sharing yields the strongest membership leakage, with ROC-AUC of 0.7537, TPR@1%FPR of 0.3936, and advantage of 0.6583. FedAvg also exposes membership information, with ROC-AUC of 0.6024. FedDyMem obtains the lowest attack performance, with ROC-AUC of 0.5310, TPR@1%FPR of 0.0288, and advantage of 0.0960, indicating reduced empirical membership leakage.

Client Domain and Property Inference Attacks. We further evaluate whether server-visible messages reveal client-level distributional information. On the texture task, let $p_{n,g}$ denote

Table 17 Client domain and property inference on the texture task under the passive honest but curious server threat model

Method	Server-visible Message	Client Domain Inference		Client Property Inference		
		Balanced Accuracy(%) ↓	Macro F1(%) ↓	AUROC ↓	Balanced Accuracy(%) ↓	F1 Score (%) ↓
FedAvg	$\Delta\boldsymbol{\theta}_n^{(T)}$	66.8 ± 5.9	64.5 ± 6.2	0.840 ± 0.050	76.1 ± 5.1	75.0 ± 5.5
Raw-memory Sharing	$\{\mathbf{M}_{n,i}^{(T)}\}_i$	94.7 ± 2.7	94.1 ± 2.9	0.985 ± 0.010	95.2 ± 2.5	95.0 ± 2.7
FedDyMem	$\mathbf{R}_n^{(T)}$	$\mathbf{54.9 \pm 6.4}$	$\mathbf{51.7 \pm 7.0}$	$\mathbf{0.720 \pm 0.060}$	$\mathbf{65.7 \pm 6.2}$	$\mathbf{64.9 \pm 6.5}$

Values are reported as mean ± standard deviation across five partition level attacker train and test splits constructed from 20 independently generated federation partition
Bold values indicate the lowest attack scores

the proportion of subtype g in client n's training data. The dominant client domain and binary client property are defined as

$$y_n^{\text{dom}} = \arg\max_g p_{n,g}, \qquad z_{n,g} = \mathbb{I}\left[p_{n,g} \geq \rho\right]. \tag{40}$$

We compare FedAvg updates, raw memory messages, and FedDyMem reduced memories using the same tokenization and Deep Sets multitask attacker Zaheer et al. (2017). Table 17 shows that raw memory sharing exposes the strongest client distribution signal, with 94.7% domain Balanced Accuracy and property AUROC of 0.985. FedAvg also reveals client-specific information, with 66.8% domain Balanced Accuracy and property AUROC of 0.840. FedDyMem achieves the lowest attack scores, reducing domain Balanced Accuracy to 54.9% and property AUROC to 0.720. These results show that memory-reduce lowers empirical client domain and property leakage relative to raw feature sharing and parameter update sharing under the evaluated threat model.

7 Conclusion

We proposed FedDyMem, a federated framework for unsupervised image anomaly detection that exchanges reduced dynamic memory banks instead of full model parameters. FedDyMem aligns heterogeneous client-side normal representations through a memory generator and metric loss, compresses local memories through memory-reduce, and aggregates multiple normal modes using server-side K-means. Experiments on six industrial and medical federated UAD tasks show that FedDyMem improves anomaly detection and localization over the evaluated federated baselines while reducing the size of the exchanged representation. The additional studies further support its robustness under the evaluated non-IID partitions, client participation changes, limited training-data contamination, and rare normal modes. *Limitations.* FedDyMem is evaluated on federations where clients share the same inspection or screening objective but have heterogeneous normal data across sources or subtypes. Its applicability to semantically different anomaly detection tasks remains unexplored. FedDyMem is specifically designed for UAD architectures with trainable memory-generation modules. Applying it to other UAD architectures would require corresponding changes to the representation-sharing and aggregation mechanisms. Although we evaluate varying client numbers and non-IID partitions, larger-scale deployments and coordinated adversarial behaviors may pose additional challenges. Memory-reduce mitigates privacy risks through representation compression, while

stronger protection against broader attack models may require secure aggregation or differential privacy.

Acknowledgements This work was supported in part by Postgraduate Research Practice Innovation Program of Jiangsu Province (Grant Number: KYCX25_0394) and the National Natural Science Foundation of China (62341408).

Funding Postgraduate Research Practice Innovation Program of Jiangsu Province (Grant Number: KYCX25_0394) and the National Natural Science Foundation of China (62341408).

Data Availability The datasets analyzed during the current study are available in the following repositories:
https://www.mvtec.com/company/research/datasets/mvtec-ad,
https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mridataset,
https://www.kaggle.com/datasets/felipekitamura/head-ct-hemorrhage,
https://www.kaggle.com/c/rsna-pneumonia-detection-challenge,
https://www.kaggle.com/c/vinbigdata-chest-xray-abnormalities-detection, and
https://huggingface.co/texture-ad.

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Almanifi, O. R. A., Chow, C. O., Tham, M. L., et al. (2023). Communication and computation efficiency in federated learning: A survey. *Internet of Things, 22*, Article 100742.

Alsulaimawi, Z. (2024). Federated learning with privacy-preserving active learning: A min–max mutual information approach. In *2024 international conference on artificial intelligence in information and communication (ICAIIC)* (pp. 503–508). IEEE.

Bae, J., Lee, J. H., & Kim, S. (2023). Pni: Industrial anomaly detection using position and neighborhood information. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6373–6383).

Bai, H., Mou, S., & Likhomanenko, T. (2023). Vision datasets: A benchmark for vision-based industrial inspection arXiv:2306.07890 arXiv preprint.

Baid, U., Ghodasara, S., & Mohan, S. (2021). The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification arXiv:2107.02314 arXiv preprint.

Bergmann, P., Fauser, M., & Sattlegger, D. (2019). Mvtec ad-a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9592–9600).

Cai, Y., Zhang, W., & Chen, H. (2024). Medianomaly: A comparative study of anomaly detection in medical images arXiv:2404.04518 arXiv preprint.

Cao, S., Liu, S., Yang, Y., et al. (2025). A hybrid and efficient federated learning for privacy preservation in IoT devices. *Ad Hoc Networks, 170*, Article 103761.

Cao, Y., Wang, Y., Yang, J., et al. (2025). AOI-OPEN: Federated operation and control for DAO-based trustworthy and intelligent AOI ecology. *Frontiers of Information Technology & Electronic Engineering, 26*(7), 1209–1221.

Cohen, N., & Hoshen, Y. (2020). Sub-image anomaly detection with deep pyramid correspondences arXiv:2005.02357 arXiv preprint.

Defard, T., Setkov, A., & Loesch, A. (2021). Padim: A patch distribution modeling framework for anomaly detection and localization. In *International conference on pattern recognition* (pp. 475–489). Springer.

Dosovitskiy, A., & Brox, T. (2016). Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4829–4837).

Fyrbiak, M., Strauß, S., & Kison, C. et al (2017). Hardware reverse engineering: Overview and open challenges. In: *2017 IEEE 2nd international verification and security workshop (IVSW)* (pp. 88–94). IEEE.

Han, S., Park, S., & Wu, F. (2022). Fedx: Unsupervised federated learning with cross knowledge distillation. In *European conference on computer vision* (pp. 691–707). Springer.

He, H., Bai, Y., Zhang, J., et al. (2024). Mambaad: Exploring state space models for multi-class unsupervised anomaly detection. *Advances in Neural Information Processing Systems, 37*, 71162–71187.

He, K., Zhang, X., & Ren, S. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, Z., Zhang, T., & Lee, R. B. (2019). Model inversion attacks against collaborative inference. In *Proceedings of the 35th annual computer security applications conference* (pp. 148–162).

Jezek, S., Jonak, M., & Burget, R. (2021). Deep learning-based defect detection of metal parts: evaluating current methods in complex conditions. In *2021 13th International congress on ultra modern telecommunications and control systems and workshops (ICUMT)* (pp. 66–71). IEEE.

Karimireddy, S. P., Kale, S., & Mohri, M. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning* (pp. 5132–5143). PMLR.

Kermany, D. S., Goldbaum, M., & Cai, W. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell, 172*(5), 1122–1131.

Kim, H., Kwak, Y., & Jung, M. (2023). Protofl: Unsupervised federated learning via prototypical distillation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6470–6479).

Lee, J. C., Kim, T., & Park, E. (2025). Continuous memory representation for anomaly detection. In *European conference on computer vision* (pp. 438–454). Springer.

Lee, S., Lee, S., & Song, B. C. (2022). Cfa: Coupled-hypersphere-based feature adaptation for target-oriented anomaly localization. *IEEE Access, 10*, 78446–78454.

Lei, T., Chen, S., & Wang, B. (2025). Adapted-moe: Mixture of experts with test-time adaption for anomaly detection. In *International conference on intelligent computing* (pp. 427–441). Springer.

Li, J., Li, C., Fan, J., et al. (2023). Online distributed stochastic gradient algorithm for non-convex optimization with compressed communication. *IEEE Transactions on Automatic Control, 69*(2), 936–951.

Li, T., Sahu, A. K., Talwalkar, A., et al. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine, 37*(3), 50–60.

Li, T., Sahu, A. K., Zaheer, M., et al. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems, 2*, 429–450.

Li, X., Jiang, M., & Zhang, X. et al (2021a). Fedbn: Federated learning on non-iid features via local batch normalization. arXiv preprint arXiv:2102.07623

Li, X. C., Zhan, D. C., & Shao, Y. et al (2021b). Fedphp: Federated personalization with inherited private models. In: *Joint European conference on machine learning and knowledge discovery in databases* (pp 587–602). Springer.

Li, Z., Lin, Z., Shao, J., et al. (2024a). Fedcir: Client-invariant representation learning for federated non-iid features. *IEEE Transactions on Mobile Computing,23*(11), 10509–10522.

Li, Z., Sun, Y., Shao, J., et al. (2024). Feature matching data synthesis for non-iid federated learning. *IEEE Transactions on Mobile Computing, 23*(10), 9352–9367.

Liang, Y., Zhang, J., & Zhao, S. (2023). Omni-frequency channel-selection representations for unsupervised anomaly detection. *IEEE Transactions on Image Processing, 32*, 4327–4340.

Liao, X., Liu, W., & Chen, C. (2024). Rethinking the representation in federated unsupervised learning with non-iid data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 22841–22850).

Liu, R., Lehman, J., & Molino, P. (2018). An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in neural information processing systems* (p. 31).

Liu, Z., Zhou, Y., & Xu, Y. (2023). Simplenet: A simple network for image anomaly detection and localization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 20402–20411).

Luo, J., & Wu, S. (2022). Adapt to adaptation: Learning personalization for cross-silo federated learning. In *IJCAI: Proceedings of the conference* (p. 2166). NIH Public Access.

Luo, W., Cao, Y., & Yao, H. (2025). Exploring intrinsic normal prototypes within a single image for universal anomaly detection. In *Proceedings of the computer vision and pattern recognition conference* (pp. 9974–9983).

Lvd, M., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research, 9*, 2579–2605.

Macpherson, M. S., Hutchinson, C. E., Horst, C., et al. (2023). Patient reidentification from chest radiographs: An interpretable deep metric learning approach and its applications. *Radiology: Artificial Intelligence, 5*(6), Article e230019.

McMahan, B., Moore, E., & Ramage, D., et al. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282). PMLR.

Nguyen, D. C., Ding, M., Pathirana, P. N., et al. (2021). Federated learning for covid-19 detection with generative adversarial networks in edge cloud computing. *IEEE Internet of Things Journal, 9*(12), 10257–10271.

Nguyen, H. Q., Lam, K., Le, L. T., et al. (2022). Vindr-cxr: An open dataset of chest x-rays with radiologist's annotations. *Scientific Data, 9*(1), 429.

Roth, K., Pemula, L., & Zepeda, J. (2022). Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14318–14328).

Saha, S., Hota, A., Chattopadhyay, A. K., et al. (2024). A multifaceted survey on privacy preservation of federated learning: progress, challenges, and opportunities. *Artificial Intelligence Review, 57*(7), 184.

Salehi, M., Sadjadi, N., & Baselizadeh, S. (2021). Multiresolution knowledge distillation for anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14902–14912).

Shlezinger, N., Chen, M., Eldar, Y. C., et al. (2020). Uveqfed: Universal vector quantization for federated learning. *IEEE Transactions on Signal Processing, 69*, 500–514.

Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105–6114) PMLR.

Tan, Q., Li, Q., & Zhao, Y. (2024). Defending against data reconstruction attacks in federated learning: An information theory approach. In *33rd USENIX security symposium (USENIX Security 24)* (pp. 325–342).

Tan, Y., Long, G., & Liu, L. et al (2022a). Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 8432–8440).

Tan, Y., Long, G., Ma, J., et al. (2022). Federated learning from pre-trained models: A contrastive learning approach. *Advances in Neural Information Processing Systems, 35*, 19332–19344.

Wang, X., Peng, Y., & Lu, L. (2017). Chestx-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2097–2106).

Xie, G., Wang, J., & Liu, J. (2024). Im-iad: Industrial image anomaly detection benchmark in manufacturing. *IEEE Transactions on Cybernetics, 54*(5), 2720–2733.

Xu, J., & Wang, H. (2020). Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective. *IEEE Transactions on Wireless Communications,20*(2), 1188–1200.

Xu, X., Yang, M., & Yi, W. (2024). A stealthy wrongdoer: Feature-oriented reconstruction attack against split learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12130–12139).

Yang, Z., Bao, W., Yuan, D., et al. (2022). Federated learning with nesterov accelerated gradient. *IEEE Transactions on Parallel and Distributed Systems, 33*(12), 4863–4873.

You, Z., Cui, L., Shen, Y., et al. (2022). A unified model for multi-class anomaly detection. *Advances in Neural Information Processing Systems, 35*, 4571–4584.

Zagoruyko, S., & Komodakis, N. (2016). *Wide residual networks. British Machine Vision Conference*. British Machine Vision Association.

Zaheer, M., Kottur, S., & Ravanbakhsh, S. (2017). Deep sets. *Advances in Neural Information Processing Systems, 30*, 3391–3401.

Zhang, C., Xie, Y., Chen, T., et al. (2024). Prototype similarity distillation for communication-efficient federated unsupervised representation learning. *IEEE Transactions on Knowledge and Data Engineering, 36*(11), 6865–6876.

Zhang, H., Li, C., & Dai, W. (2023). Fedcr: Personalized federated learning based on across-client common representation with conditional mutual information regularization. In *International conference on machine learning* (pp. 41314–41330). PMLR.

Zhang, J., Hua, Y., Cao, J., et al. (2024). Eliminating domain bias for federated learning in representation space. *Advances in Neural Information Processing Systems, 36*, 14204–14227.

Zhang, R., Li, H., & Tian, L. et al (2024c). Vertical federated learning across heterogeneous regions for industry 4.0. *IEEE Transactions on Industrial Informatics*, *20*(8), 10145–10155.

Zhao, Y., Liu, Q., Liu, P., et al. (2024). Medical federated model with mixture of personalized and shared components. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 47*(1), 433–449.

Zhuang, W., Gan, X., & Wen, Y. (2021). Collaborative unsupervised visual representation learning from decentralized data. In *2021 IEEE/CVF International Conference on Computer Vision* (pp. 4892–4901). IEEE.

Zou, Y., Jeong, J., & Pemula, L. (2022). Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. In *European conference on computer vision* (pp. 392–408). Springer.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.