

Fed-BRDECS: Privacy-Preserving and Heterogeneity-Aware Federated Deep Embedded Clustering

Haemin Park^{1*}, Diego Klabjan¹, Martin W. Braun², Xiuqi Li², Balakrishnan Ananthanarayanan²

¹Department of Industrial Engineering & Management Sciences, Northwestern University, USA

²Intel Corporate, USA

{haemin.park1,d-klabjan}@northwestern.edu

{martin.w.braun, xiuqi.li, a.bala.krishnan}@intel.com

Abstract

Federated deep clustering seeks to learn clustering-friendly representations from decentralized unlabeled data while preserving client privacy. However, Deep Embedded Clustering (DEC)-style objectives depend on global soft-assignment statistics that require clients to reveal their sensitive information. We propose Fed-BRDECS, a privacy-preserving and heterogeneity-aware federated deep embedded clustering framework. Fed-BRDECS replaces the globally normalized clustering objective with a locally computable sample-stability loss, avoiding the transmission of local soft-assignment distributions. To tackle non-IID client distributions, we introduce prediction-balanced sampling, which oversamples locally rare predicted clusters without requiring ground-truth labels, and centroid-level restarting, which periodically refreshes biased or inactive centroids. Experiments on image and text clustering benchmarks show that Fed-BRDECS consistently outperforms representative federated clustering and deep clustering baselines under both IID and non-IID partitions. We further demonstrate its applicability to federated time-series anomaly detection, where it improves reconstruction-based detectors without adding inference-time cost.

1. Introduction

Clustering is a fundamental unsupervised learning problem with broad applications in healthcare, industrial Internet of Things, mobile sensing, network monitoring, and large-scale user behavior analysis (Jain 2010; Rieke et al. 2020; Nguyen et al. 2021; Pfeifer et al. 2024; Presotto, Civitarese, and Bettini 2022). In many such domains, however, data are naturally distributed across institutions, devices, or edge systems and cannot be centrally collected due to privacy, ownership, regulatory, or communication constraints. Federated learning (FL) provides a natural paradigm for this setting by enabling multiple clients to collaboratively train a model while keeping raw data local (McMahan et al. 2017). Extending clustering to the federated setting is therefore important for privacy-preserving unsupervised representation learning, where labels are unavailable and data heterogeneity is often severe (Dennis, Li, and Smith 2021; Garst and Reinders 2024). However, directly applying centralized deep clustering methods to FL is nontrivial.

Deep Embedded Clustering (Xie, Girshick, and Farhadi 2016) and its improved variant IDEC (Guo et al. 2017) are the dominant modern approaches to deep clustering, and both are strong in the centralized regime. After autoencoder pre-training, they jointly refine latent representations and cluster assignments using soft assignments and an auxiliary target distribution. While effective in centralized learning, this design creates a fundamental tension in FL: the auxiliary target depends on dataset-level assignment frequencies. In FL, reproducing it requires either exchanging local cluster statistics, which exposes client composition, or constructing targets from incomplete local views, which can bias optimization under heterogeneity. This is not the only obstacle.

The second challenge is heterogeneity. In federated clustering, each client may observe only a subset of the global cluster structure, and local cluster proportions can differ substantially from the global distribution (Dennis, Li, and Smith 2021; Pan et al. 2023). As a result, cluster-frequency-based objectives may over-emphasize locally dominant clusters and under-train rare or missing clusters. Such imbalance can bias the learned embedding space, distort centroid updates, and amplify client drift. The third challenge is cluster collapse and early saturation. Once the encoder and centroids become aligned with an imperfect early partition, subsequent training may reinforce the same assignments rather than correct them. Recent work identifies this phenomenon as a reclustering barrier, where performance saturates early and periodic reclustering alone may be insufficient to fully recover from poor initial partitions (Miklautz et al. 2025). In FL, this early saturation can be further amplified because client updates are computed from partial, heterogeneous views and then averaged at the server.

We propose Fed-BRDECS (Federated Balanced-Sampling and Restarted Deep Embedded Clustering with Sample Stability), a federated deep clustering framework that preserves the standard IDEC pipeline of autoencoder pretraining, centroid initialization, and clustering refinement, while adding three components. These enhancements address the privacy issues inherent in directly applying DEC to FL, while simultaneously tackling the core challenges of federated deep clustering.

First, to remove the dependence on global assignment statistics, Fed-BRDECS replaces the KL-divergence-based IDEC clustering objective with a locally computable sample-

*Corresponding author.

stability loss inspired by DECS (Cheng et al. 2024). Since this objective is computed using only local samples and current centroids, clients do not need to transmit soft-assignment distributions or cluster-frequency statistics, and communicate only model parameters. Second, to mitigate local imbalance under heterogeneous FL, Fed-BRDECS uses a prediction-based balanced sampler that oversamples each client’s locally rare predicted clusters using the model’s own assignments in place of cluster labels. This acts as an unsupervised form of class-balanced importance sampling, reducing the dominance of local majority clusters during client training. Third, to recover collapsed clusters and avoid early commitment, Fed-BRDECS introduces a restarting mechanism that periodically reinitializes cluster centers through federated k-means in the learned embedding space. This allows the model to revise poor assignments and correct centroid bias caused by heterogeneous client updates.

DEC has several applications, with outlier detection being one of the most common. We use Fed-BRDECS as the building block for federated outlier detection and validate on time-series data from both public benchmarks and a proprietary sensor dataset. Since many detectors are themselves built on deep embedded clustering structures (Zong et al. 2018; Obied et al. 2023), our framework can be easily applied to DEC-based or AE-based federated time-series anomaly detection algorithms. This demonstrates that the proposed framework is not limited to standard clustering benchmarks, but can also serve as a privacy-preserving representation-learning backbone for downstream unsupervised monitoring tasks.

Our contributions are summarized as follows.

1. We propose **Fed-BRDECS**, a federated deep embedded clustering framework for heterogeneous unsupervised federated learning. Unlike prior federated or distributed clustering methods that exchange local clustering information, Fed-BRDECS enables IDEC-style joint representation and clustering refinement using a locally computable sample-stability loss with privacy adherence.
2. We introduce two mechanisms to improve federated deep clustering under heterogeneous client distributions: a **prediction-based balanced sampler** that mitigates local cluster imbalance using pseudo-labels, and an **iterative restarting strategy** that refreshes centroids to reduce early commitment and cluster collapse.
3. We demonstrate consistent empirical improvements over representative federated clustering and deep clustering baselines, with ablation studies validating the contribution of stability loss, balanced sampling, and restarting.
4. We evaluate Fed-BRDECS on **federated time-series anomaly detection** as a practical stress test for unsupervised federated representation learning, showing that the proposed framework can improve reconstruction-based detectors under heterogeneous client time-series distributions without adding inference-time cost.

2. Related Work

Deep clustering. DEC (Xie, Girshick, and Farhadi 2016) is a pioneering work showing that a representation can be

Table 1: Comparison with representative federated clustering methods. A check mark indicates that the method explicitly supports the corresponding property. “Privacy-preserving” means that clients do not transmit local cluster assignments or soft-assignment matrices.

Method	Privacy-preserving	Deep clustering	Heterogeneity-aware
k-FED	✓		
DWF k-Means			
FKM			
FFCM			
F-DEC		✓	
CCFC	✓	✓	
Fed-BRDECS	✓	✓	✓

learned directly from a clustering objective. DEC first pre-trains an autoencoder and then refines the encoder and centroids using the KL-divergence loss between soft assignments and a self-generated target distribution. IDEC (Guo et al. 2017) adds a reconstruction term to preserve local structure, and DCN (Yang et al. 2017) similarly connects representation learning with k-means style clustering. A common limitation of DEC-based methods is their reliance on manually constructed pseudo-targets, which may reinforce early assignment errors and depend on dataset-level assignment statistics. DECS (Cheng et al. 2024) replaces pseudo-targets with sample-stability optimization, which we adopt to eliminate global assignment statistics. BRB (Miklautz et al. 2025) addresses early saturation through reclustering and soft parameter resets. Fed-BRDECS adapts these ideas to heterogeneous FL using local stability optimization and centroid-level restarting without parameter resets.

Federated clustering. Federated clustering has mostly focused on centroid-style algorithms over fixed features rather than deep representation learning. k-FED (Dennis, Li, and Smith 2021) is a one-shot method in which each client uploads its local cluster centers only once. FKM (Garst and Reinders 2024) extends this paradigm to multiple communication rounds. Other methods address specific limitations of federated k-means: DWF k-Means (Holzer, Jacob, and Kavane 2023) introduces dynamic weighting, Fork (Bravo and Dalleiger 2026) improves robustness, and FedDP k-Means (Scott, Lampert, and Saulpic 2025) incorporates differential privacy. In contrast to these hard-assignment approaches, FFCM (Stallmann and Wilbik 2022) adapts fuzzy c-means to federated learning, allowing samples to have soft membership across multiple clusters. However, these methods still perform centroid-based clustering on raw or precomputed features rather than jointly learning deep representations, and they also exchange explicit local cluster statistics. More recently, efforts have emerged to apply deep clustering to FL. F-DEC (Mashhadi, Sterner, and Murray 2021) directly applies DEC to FL by optimizing a KL-based clustering loss across clients, and FDEC (Xu, Liao, and Yang 2025) attempts to achieve robustness through hybrid federated averaging. CCFC (Liu, Yan, and Zhang 2024) incorporates a cluster-contrastive loss to learn clustering-friendly representations. Still, these deep frameworks do not explicitly resolve core FL challenges such as centroid collapse, data hetero-

geneity, and privacy leakage tied to transmitting local soft assignment statistics.

Imbalance in federated and unsupervised learning. Heterogeneity and imbalance are central challenges in FL. Supervised FL methods commonly address them through local data rebalancing, loss reweighting, or aggregation adjustment (Wang et al. 2021; Seol and Kim 2023; Chung and Lee 2025; Park et al. 2026). However, these methods rely on labels or label-distribution information, which are unavailable in unsupervised federated clustering. In unlabeled settings, related works—particularly in centralized environments—utilize inferred groups or pseudo-clusters to mitigate imbalance. For instance, some approaches balance unlabeled datasets using cluster IDs as pseudo-classes, while others partition majority classes into subclusters for imbalanced classification (Suayama and Nakamura 2025; Lim 2024). We adapt this concept to federated clustering. However, rather than treating predicted cluster assignments as supervised targets, we use them strictly to define a local sampling distribution paired with an oversampling strategy.

3. Method

3.1. Preliminaries: DEC/IDEC

DEC and IDEC combine representation learning with centroid-based clustering to map raw data into a semantically meaningful latent space. Let $\mathcal{D} = \{x_i\}_{i=1}^N$ be a centralized dataset. An encoder $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$ maps an input x_i to a latent representation $z_i = f_\theta(x_i) \in \mathcal{Z}$ and a decoder $g_\phi : \mathcal{Z} \rightarrow \mathcal{X}$ reconstructs the input, encouraging the latent representation to preserve local structures. The framework maintains C trainable centroids $\{\mu_j\}_{j=1}^C \subset \mathcal{Z}$. Standard IDEC operates in three stages:

1. **Autoencoder Pretraining:** The network minimizes reconstruction loss to establish a meaningful initial embedding space:

$$\min_{\phi, \theta} \mathcal{L}_{\text{rec}} = \frac{1}{N} \sum_{i=1}^N \|x_i - g_\phi(f_\theta(x_i))\|_2^2. \quad (1)$$

2. **Centroid Initialization:** centroid-based clustering (usually, k-Means++) is applied on the pretrained embeddings $\{z_i = f_\theta(x_i)\}_{i=1}^N$ to initialize the centroids $\{\mu_j\}_{j=1}^C$ where ϕ, θ are learned through (1).
3. **Joint Clustering Refinement:** The encoder, decoder, and centroids are jointly optimized. For each sample, a soft assignment q_{ij} is computed using a Student- t kernel:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|_2^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{\ell=1}^C (1 + \|z_i - \mu_\ell\|_2^2/\alpha)^{-\frac{\alpha+1}{2}}}, \quad (2)$$

where $\alpha > 0$ denotes the degree-of-freedom parameter. A large value of q_{ij} indicates that z_i is closer to centroid μ_j relative to the other centroids. An auxiliary target distribution P is then constructed to sharpen confident assignments after constructing soft cluster frequencies

$$f_j = \sum_{i=1}^N q_{ij}:$$

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{\ell=1}^C q_{i\ell}^2/f_\ell}. \quad (3)$$

The parameters are updated via stochastic gradient descent by minimizing the joint loss:

$$\mathcal{L}_{\text{IDEC}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{clu}} \text{KL}(P\|Q), \quad (4)$$

where λ_{clu} balances reconstruction and cluster refinement. DEC only uses clustering loss without reconstruction loss for clustering refinement.

3.2. Problem Formulation: Federated Deep Embedded Clustering

We consider an unsupervised federated learning setting with K clients. Client k owns an unlabeled private dataset $\mathcal{D}_k = \{x_{k,i}\}_{i=1}^{n_k}$. The global dataset is defined as $\mathcal{D} = \bigcup_{k=1}^K \mathcal{D}_k$ and the total number of samples is $N = \sum_{k=1}^K n_k$. The goal is to learn a single global clustering model that partitions the distributed dataset into C clusters.

Privacy Issue in FL. Directly applying IDEC to FL introduces a severe problem. Computing the global target distribution requires aggregating dataset-wide soft frequencies f_j . In FL, this forces each client k to compute and upload local soft counts $f_{k,j} = \sum_{x_i \in \mathcal{D}_k} q_{ij}$ to the server. This mechanism leaks sensitive information regarding local cluster compositions and relies on stale global statistics between periodic server refreshes. These limitations motivate Fed-BRDECS to replace the global KL target with a locally computable sample-stability objective.

Problem Formulation Following the deep embedded clustering framework, the global model is denoted by

$$\Theta = \{\theta, \phi, \mu_1, \dots, \mu_C\}.$$

It consists of an encoder $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$, a decoder $g_\phi : \mathcal{Z} \rightarrow \mathcal{X}$, and C centroids $\{\mu_j\}_{j=1}^C \subset \mathcal{Z}$ in the latent space $\mathcal{Z}$. Given a trained model, the cluster assignment of a sample x is determined by the nearest centroid in the latent space:

$$\hat{y}(x) = \arg \min_{j \in \{1, \dots, C\}} \|f_\theta(x) - \mu_j\|_2^2.$$

Since training is unsupervised, these assignments are model-induced pseudo-cluster labels rather than ground-truth labels. Because these local assignments can still leak privacy-sensitive information, we focus on a federated clustering design that entirely eliminates the transmission of local assignment statistics.

The federated objective can be written as the weighted average of local unsupervised objectives:

$$\min_{\Theta} \mathcal{F}(\Theta) = \sum_{k=1}^K r_k \mathcal{F}_k(\Theta; \mathcal{D}_k),$$

where $\mathcal{F}_k$ is the local clustering objective evaluated on client k 's data, and $r_k = \frac{n_k}{N}$.

This formulation highlights the central difficulty of federated deep embedded clustering. The model must learn both the representation and the centroids from decentralized, unlabeled, and potentially non-IID data, while avoiding reliance on centralized assignment statistics. Fed-BRDECS addresses this difficulty by replacing the globally normalized IDEC target objective with a local sample-stability loss, correcting local prediction imbalance through a prediction-based balanced sampler, and periodically reinitializing centroids through federated restarting.

3.3. Algorithm: Fed-BRDECS

Fed-BRDECS optimizes a federated deep clustering objective in three stages: federated representation pretraining, federated centroid initialization, and restarted clustering refinement. The first two stages follow the standard deep embedded clustering pipeline, while the refinement stage replaces IDEC’s globally normalized target objective with local sample stability, prediction-balanced sampling, and periodic federated restarting.

Federated representation pretraining. In the pretraining stage, each client optimizes the reconstruction loss

$$\mathcal{L}_{\text{rec}}^{(k)} = \frac{1}{n_k} \sum_{x \in \mathcal{D}_k} \|x - g_\phi(f_\theta(x))\|_2^2$$

on its local data. The server aggregates client updates with FedAvg for T_{pre} rounds. This stage learns an initial encoder-decoder pair (f_θ, g_ϕ) that preserves the local structure of each client’s data in the latent space.

Federated centroid initialization. After pretraining, the server initializes the latent centroids using a federated clustering routine on the client embeddings:

$$\{\mu_j\}_{j=1}^C \leftarrow \text{k-FED}(\{f_\theta(\mathcal{D}_k)\}_{k=1}^K).$$

FL clustering algorithm k-FED (Dennis, Li, and Smith 2021) plays the role of centralized k-Means initialization in DEC/IDEC, but it is performed without collecting raw data or local assignment information at the server. In our implementation, each client computes local centroids from its embedded data and sends only these centroids to the server. The server then aggregates the local centroids to obtain the initial global centroids.

Restarted clustering refinement. The refinement stage is organized into R restart iterations. Unlike BRB-style reclustering, Fed-BRDECS does not apply a soft reset to the model parameters. Instead, each restart performs a centroid-level federated reclustering step while keeping the representation network continuous across restarts. This design focuses on correcting centroid degeneration, inactive clusters, and heterogeneity-induced assignment imbalance.

At the beginning of restart r , the server simply recomputes the centroids by running k-FED on the current client embeddings:

$$\{\mu_j\}_{j=1}^C \leftarrow \text{k-FED}(\{f_\theta(\mathcal{D}_k)\}_{k=1}^K),$$

warm-started from the previous centroids. This federated reclustering step relocates centroids to occupied regions of

the current embedding space and can revive centroids that have become inactive under non-IID local updates.

Within each restart iteration, Fed-BRDECS performs T rounds of federated refinement. On client k , soft assignments are computed from the local model using the Student- t kernel with parameter α :

$$q_{kij} = \frac{(1 + \|z_{k,i} - \mu_{k,j}\|_2^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{\ell=1}^C (1 + \|z_{k,i} - \mu_{k,\ell}\|_2^2/\alpha)^{-\frac{\alpha+1}{2}}}, \quad z_{k,i} = f_{\theta_k}(x_{k,i}).$$

Unlike IDEC, Fed-BRDECS does not construct a target distribution from globally normalized cluster frequencies. Instead, each client evaluates sample stability, inspired by Cheng et al. (2024), from its local assignment probabilities.

Let $\tau_k \in (0, 1)$ be the threshold selected by Otsu’s method (Otsu 1979) on the assignment probabilities observed at client k . We define determinacy score $d_{kij} = ((q_{kij} - \tau_k)/\tau_k)^2$ if $q_{kij} < \tau_k$, and $d_{kij} = ((q_{kij} - \tau_k)/(1 - \tau_k))^2$ otherwise.

For sample $x_{k,i}$, we define the average determinacy as $\bar{d}_{k,i} = \frac{1}{C} \sum_{j=1}^C d_{kij}$. Then the stability score is

$$s_{k,i} = \bar{d}_{k,i} - \frac{\lambda_{\text{var}}}{C} \sum_{j=1}^C (d_{kij} - \bar{d}_{k,i})^2,$$

where λ_{var} is a proportionality coefficient. The first term rewards assignment determinacy with respect to all centroids, while the second term regularizes the dispersion of determinacy values across centroids. The local stability loss over a mini-batch $\mathcal{B}_k$ is

$$\mathcal{L}_{\text{stab}}^{(k)} = 1 - \frac{1}{|\mathcal{B}_k|} \sum_{x_{k,i} \in \mathcal{B}_k} s_{k,i}.$$

The local refinement objective is

$$\mathcal{L}_k = \mathcal{L}_{\text{rec}}^{(k)} + \lambda_{\text{stab}} \mathcal{L}_{\text{stab}}^{(k)},$$

where λ_{stab} controls the contribution of the stability loss. The reconstruction term is also retained during refinement, as in IDEC, to preserve local structure in the embedding space. The encoder, decoder, and centroids are updated jointly by local SGD between restart steps.

Prediction-balanced sampler. To address local prediction imbalance, each client periodically updates a prediction-based sampling distribution. Every T_{samp} rounds, client k computes local soft cluster frequencies $f_{k,j} = \sum_{i=1}^{n_k} q_{kij}$ ($j = 1, \dots, C$) and hard predicted assignments $\hat{y}_{k,i} = \arg \max_{j \in \{1, \dots, C\}} q_{kij}$.

The inverse predicted-cluster frequency is $\tilde{w}_{k,j} = \frac{1}{f_{k,j} + \epsilon}$, and each sample receives weight

$$w_{k,i} = \frac{\tilde{w}_{k,\hat{y}_{k,i}}}{\sum_{\ell=1}^C \tilde{w}_{k,\ell}}.$$

Mini-batches are then sampled with replacement according to probabilities proportional to $w_{k,i}$. Thus, samples assigned to locally rare predicted clusters are sampled more often,

while samples assigned to locally dominant predicted clusters are downsampled. The predicted assignments are used only to define the sampling distribution and are not treated as supervised labels.

At refinement round t of restart iteration r , the server broadcasts the current global model $\Theta^{r,t-1}$ to the clients. Each client performs E local steps on $\mathcal{L}_k$ using the current balanced sampler and uploads its updated parameters $\Theta_{k,E}^{r,t}$. The server aggregates the returned models as

$$\Theta^{r,t} = \sum_k r_k \Theta_{k,E}^{r,t}.$$

4. Experiments

4.1. Setup for Clustering on Public Datasets

Datasets. We evaluate Fed-BRDECS on four standard deep clustering benchmarks: MNIST (Deng 2012), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), KMNIST (Clanuwat et al. 2018), and REUTERS-10K (Lewis et al. 2004). The three image datasets each contain ten ground-truth categories, while REUTERS-10K is a four-category text clustering benchmark commonly used in DEC/IDEC evaluations. We follow the standard preprocessing protocol used in deep embedded clustering methods (Xie, Girshick, and Farhadi 2016; Guo et al. 2017; Cheng et al. 2024). Ground-truth labels are used only for federated distribution of samples to clients (partitioning) and final evaluation, and are never used during unsupervised training.

Federated partitioning. All experiments are conducted with $K = 10$ clients. For each dataset, we evaluate two federation settings. In the IID setting, samples are uniformly distributed across clients. In the non-IID setting, client data distributions are generated using a Dirichlet partition with concentration parameter $\alpha = 0.5$, as described in Hsu, Qi, and Brown (2019). The non-IID setting tests whether each method can learn meaningful global clusters when clients observe different local cluster proportions.

Baselines. We compare Fed-BRDECS with three groups of baselines. The first group consists of federated centroid clustering methods over fixed features: k-FED (Dennis, Li, and Smith 2021), FKM (Garst and Reinders 2024), and DWF k-Means (Holzer, Jacob, and Kavane 2023). These methods do not learn deep representations.

The second group consists of deep federated clustering baselines. Fed-AE first trains an autoencoder in a federated manner and then applies k-FED to the learned embeddings, isolating the effect of representation pretraining without joint clustering refinement. F-DEC (Mashhadi, Sterner, and Murray 2021) is a direct federated implementation of IDEC, where clients optimize IDEC reconstruction and the KL clustering objective under FedAvg. This baseline uses the same autoencoder backbone but retains the DEC/IDEC target distribution objective, therefore, requires exchange of local clustering information to construct the global target distribution. CCFC (Liu, Yan, and Zhang 2024) is a recent contrastive federated deep clustering method.

Algorithm 1: Fed-BRDECS

Require: Client datasets $\{\mathcal{D}_k\}_{k=1}^K$, number of clusters C , pretraining rounds T_{pre} , restart iterations R , refinement rounds T , local steps E , sampler period T_{samp} , loss weights λ_{stab} , stability parameter λ_{var}

Ensure: Global model $\Theta = \{\theta, \phi, \mu_1, \dots, \mu_C\}$

(Step 1) Federated Representation Pretraining.

- 1: Server initializes the global model Θ^0
- 2: **for** $t = 1, \dots, T_{\text{pre}}$ **do**
- 3: Server broadcasts Θ^{t-1} to clients.
- 4: **for** $k = 1, \dots, K$ **do**
- 5: Initialize local model $\Theta_k^{t,0} \leftarrow \Theta^{t-1}$.
- 6: Perform E local SGD steps on $\mathcal{L}_k = \mathcal{L}_{\text{rec}}^{(k)}$.
- 7: Upload $\Theta_k^{t,E}$ to the server.
- 8: **end for**
- 9: Server aggregates: $\Theta^t = \sum_k r_k \Theta_k^{t,E}$.
- 10: **end for**

(Step 2) Federated Centroid Initialization.

- 11: Perform centroid initialization using k-FED:

$$\{\mu_j\}_{j=1}^C \leftarrow \text{k-FED}(\{f_{\theta^{T_{\text{pre}}}}(\mathcal{D}_k)\}_{k=1}^K),$$

and update $\Theta^{T_{\text{pre}}}$ with these centroids.

(Step 3) Restarted Clustering Refinement.

- 12: Initialize $\Theta^{0,0} \leftarrow \Theta^{T_{\text{pre}}}$.
 - 13: **for** $r = 1, \dots, R$ **do**
 - 14: Initialize $\Theta^{r,0} \leftarrow \Theta^{r-1,T}$.
 - 15: Recompute centroids:
 - 16: $\{\mu_j^r\}_{j=1}^C \leftarrow \text{k-FED}(\{f_{\theta^{r,0}}(\mathcal{D}_k)\}_{k=1}^K)$.
 - 17: Set $\Theta^{r,0} \leftarrow \{\theta^{r,0}, \phi^{r,0}, \mu_1^r, \dots, \mu_C^r\}$.
 - 18: **for** $t = 1, \dots, T$ **do**
 - 19: Server broadcasts $\Theta^{r,t-1}$ to clients.
 - 20: **for** $k = 1, \dots, K$ **do**
 - 21: Initialize local model $\Theta_{k,0}^{r,t} \leftarrow \Theta^{r,t-1}$.
 - 22: **if** $t \equiv 0 \pmod{T_{\text{samp}}}$ **then**
 - 23: Compute sampler weights $w_{k,i}$.
 - 24: **end if**
 - 25: Perform E local SGD steps on
 - 26: $\mathcal{L}_k = \mathcal{L}_{\text{rec}}^{(k)} + \lambda_{\text{stab}} \mathcal{L}_{\text{stab}}^{(k)}$
 - 27: using prediction-balanced mini-batches.
 - 28: Upload $\Theta_{k,E}^{r,t}$ to the server.
 - 29: **end for**
 - 30: Server aggregates: $\Theta^{r,t} = \sum_k r_k \Theta_{k,E}^{r,t}$.
 - 31: **end for**
 - 32: **end for**
 - 33: **return** $\Theta^{R,T}$
-

The third group consists of our ablation variants. Fed-DECS uses only the local sample-stability loss, corresponding to the stability component (denoted as S). Fed-BDECS adds prediction-balanced sampling ($S+B$). Fed-BRDECS is the full method with sample stability, prediction-balanced

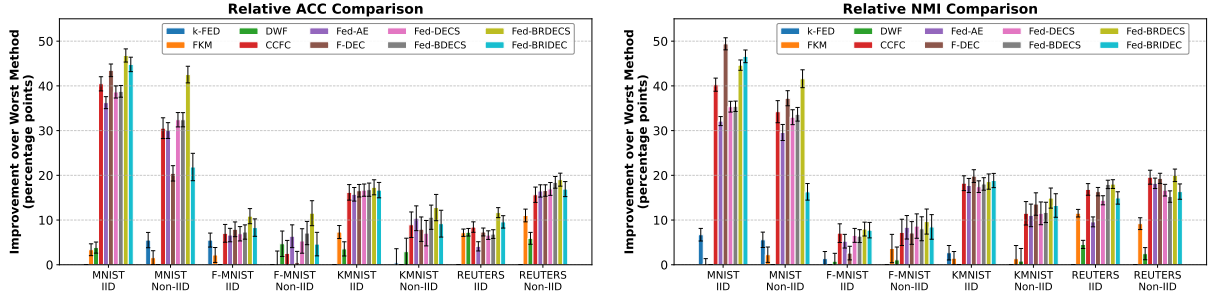


Figure 1: Performance gains relative to the worst-performing method for each dataset and partition. Bars show percentage-point improvements in ACC (left) and NMI (right), and error bars indicate one standard deviation over five runs. Higher is better. Lower performance of DEC-based refinement compared with Fed-AE indicates that the refinement stage can degrade the pretrained representation, especially under non-IID client distributions.

sampling, and centroid-level restarting ($S+B+R$). We also report Fed-BRIDECS, which applies balanced sampling and restarting to the IDEC-style objective, to separate the effect of our stability objective from the effect of the balancing and restarting mechanisms.

Evaluation metric. All clustering methods are evaluated by clustering accuracy (ACC) and Normalized Mutual Information (NMI), which are widely used in unsupervised clustering. Unless otherwise stated, we report the average over five independent runs with different random seeds.

Implementation details. For a fair comparison, all deep clustering methods use the same standard DEC autoencoder architecture adopted in prior work (Xie, Girshick, and Farhadi 2016; Guo et al. 2017). The encoder follows the standard DEC-style multilayer perceptron architecture with a low-dimensional latent embedding, and the decoder is symmetric. Autoencoder pretraining is performed by FedAvg, after which centroids are initialized using k-FED on the client embeddings. During refinement, clients perform local SGD steps on their respective objectives and the server aggregates client models by FedAvg. Extended implementation details are provided in Appendix B.

4.2. Clustering Results

In Figure 1, results are shown as percentage-point improvements over the worst-performing method for each dataset, partition, and metric. Fed-BRDECS attains the highest ACC in all eight settings. The corresponding absolute ACC and NMI values are reported in Appendix B.3. Overall, Fed-BRDECS achieves the strongest and most consistent clustering performance across datasets and partition settings. Its advantage is particularly evident under non-IID distributions, indicating improved robustness to heterogeneous client data.

Effect of representation learning. The fixed-feature federated k-means methods are consistently weaker than deep clustering methods on most benchmarks (Figure 1). For example, on MNIST-IID, the best fixed-feature baseline obtains 0.4914, whereas Fed-AE reaches 0.8155. This confirms that learning a clustering-friendly representation remains crucial in federated clustering. However, Fed-AE is still weaker than

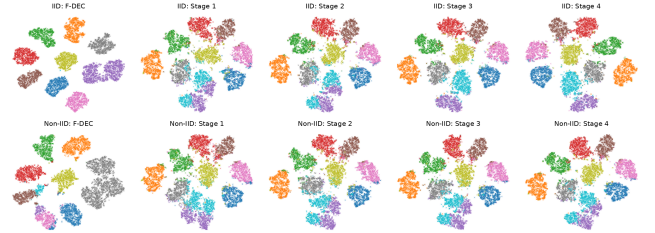


Figure 2: t-SNE visualization of learned MNIST embeddings under IID and non-IID federated partitions. The first column shows F-DEC, while the remaining columns show Fed-BRDECS across restart/refinement stages. Under non-IID partition, F-DEC exhibits clustering collapse, with two classes mixed into poorly separated regions. Fed-BRDECS mitigates this collapse through centroid-level restarting, progressively producing more compact and separated cluster structures.

Fed-BRDECS, showing that autoencoder pretraining alone is insufficient without clustering-aware refinement. In several settings, particularly under non-IID partitions, F-DEC also underperforms Fed-AE, indicating that DEC-style refinement can degrade the pretrained representation.

Effectiveness of proposed components. F-DEC is a direct federated implementation of the IDEC objective. Fed-BRDECS improves over F-DEC by 6.47 percentage points in average ACC across the eight settings, with especially large gains under non-IID partitions. Further, the ablation ladder Fed-DECS $\rightarrow$ Fed-BDECS $\rightarrow$ Fed-BRDECS shows the contribution of each component. Fed-DECS, which uses only the stability loss, already performs competitively with F-DEC while avoiding the transmission of local cluster-frequency statistics. Adding prediction-balanced sampling and centroid-level restarting further improves ACC, indicating that our B+R components help correct biased or inactive centroids during federated refinement.

Robustness to heterogeneity. Fed-BRDECS is particularly effective under non-IID partitions. Averaged over the four non-IID settings, Fed-BRDECS obtains 0.6889 ACC,

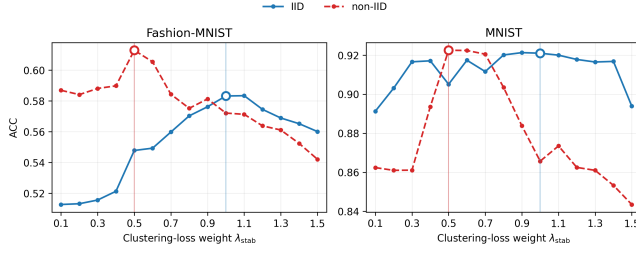


Figure 3: Sensitivity of Fed-BRDECS to the clustering-loss weight λ_{stab} on Fashion-MNIST and MNIST.

outperforming other algorithms. This suggests that the combination of prediction-balanced sampling and centroid restarting improves robustness to heterogeneous local cluster proportions.

Embedding visualization. Figure 2 visualizes the learned MNIST embeddings using t-SNE. Compared with F-DEC, Fed-BRDECS mitigates cluster collapse and cluster mix-up. The improvement is especially visible under the non-IID partition, where the baseline exhibits stronger cluster mixing, whereas Fed-BRDECS progressively separates clusters through prediction-balanced refinement and centroid-level restarting.

Ablation on the clustering-loss weight. Figure 3 examines the effect of λ_{clu} on MNIST and Fashion-MNIST. Under non-IID partitions, both datasets perform best at $\lambda_{clu} = 0.5$, whereas the IID results remain strong around $\lambda_{clu} = 1.0$. We therefore use $\lambda_{clu} = 0.5$ for non-IID partitions and $\lambda_{clu} = 1.0$ for IID partitions in the main experiments.

4.3. Application to Federated Time-Series Anomaly Detection

Beyond standard clustering benchmarks, we evaluate Fed-BRDECS in federated time-series anomaly detection (TSAD), which provides a practical stress test for unsupervised federated representation learning. Many TSAD methods are reconstruction based. A model is trained to reconstruct the normal time-series and anomalous time-series are identified by a large reconstruction error (Malhotra et al. 2016; Audibert et al. 2020; Tuli, Casale, and Jennings 2022). In federated TSAD, different clients may correspond to different machines or operating environments, so normal patterns can be highly heterogeneous. As a result, a single reconstruction model trained by FedAvg may underrepresent rare normal regimes and produce false alarms.

Applying Fed-BRDECS to reconstruction-based TSAD. Fed-BRDECS can be applied to reconstruction-based TSAD backbones as a training-time module. If the detector already includes a latent clustering component, we directly replace its clustering objective with our sample-stability loss and add prediction-balanced sampling and centroid restarting. If the detector is purely reconstruction-based, we attach a temporary projection head and centroids to the latent representation, optimize the sample-stability objective jointly

Table 2: Federated time-series anomaly detection results. “Baseline” denotes the backbone trained with FedAvg, while “+BRDECS” additionally uses Fed-BRDECS during training.

Dataset	Backbone	Baseline		+BRDECS	
		AUC-ROC	F1	AUC-ROC	F1
SMD	LSTM-AE	0.5915	0.1413	0.6038	0.1422
SMD	TranAD	0.6335	0.1829	0.6396	0.1848
PSM	LSTM-AE	0.6063	0.4415	0.6121	0.4433
PSM	TranAD	0.5792	0.4451	0.5863	0.4425
Sensor	CoFormer-AE	0.5258	0.4503	0.5544	0.4629

with reconstruction, and use the resulting pseudo-clusters for balanced sampling.

We instantiate this mechanism on LSTM-AE (Garg et al. 2021) and TranAD (Tuli, Casale, and Jennings 2022) for regularly sampled public benchmarks. For LSTM-AE, the clustering head is attached to the encoder final hidden state. For TranAD, it is attached to the latent representation used by the reconstruction module. The proprietary sensor data are irregularly sampled multivariate time series. We therefore use CoFormer-AE (Wei et al. 2023), which replaces the LSTM encoder with CoFormer. We do not evaluate TranAD or other architecture-specific TSAD variants on this dataset because they are not directly compatible with the CoFormer backbone. At inference time, the clustering head is discarded and outlier scores are computed using the original reconstruction error, so Fed-BRDECS adds no inference-time cost.

Experimental setup. We evaluate on two public multivariate TSAD benchmarks, SMD (Su et al. 2019) and PSM (Abdulaal, Liu, and Lancewicki 2021), following the federated evaluation setting of FedTADBench (Liu et al. 2022). For these public datasets, we compare LSTM-AE and TranAD trained with FedAvg against the same backbones trained with Fed-BRDECS. We also evaluate on a proprietary irregularly sampled sensor dataset collected from multiple industrial machines. For this dataset, we use CoFormer-AE as the reconstruction backbone. We report AUC-ROC and F1. Further dataset and implementation details are provided in Appendix C.2.

Results. Table 2 shows that adding Fed-BRDECS improves AUC-ROC in all five evaluated dataset-backbone settings and improves F1 in four of five settings. These gains are modest but consistent, which is expected because Fed-BRDECS is used only as a training-time representation module while clustering components along with balanced sampler might help mitigate heterogeneity under FL.

Overall, TSAD experiments show that Fed-BRDECS is not limited to standard clustering benchmarks. It can also be used as a federated representation-learning module for downstream unsupervised monitoring tasks, improving reconstruction-based outlier detectors without changing their inference procedure.

References

- Abdulaal, A.; Liu, Z.; and Lancewicki, T. 2021. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2485–2494.
- Audibert, J.; Michiardi, P.; Guyard, F.; Marti, S.; and Zuluaga, M. A. 2020. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Bravo, M.; and Dalleiger, S. 2026. Robust Federated Clustering under Heterogeneity and Adversaries. In *The 29th International Conference on Artificial Intelligence and Statistics*.
- Cheng, Z.; Li, F.; Wang, J.; and Qian, Y. 2024. Deep embedding clustering driven by sample stability. *arXiv preprint arXiv:2401.15989*.
- Chung, H.; and Lee, J. S. 2025. Multi-level analyzation of imbalance to resolve non-IID-ness in federated learning. *Neurocomputing*, 626: 129528.
- Clanuwat, T.; Bober-Irizar, M.; Kitamoto, A.; Lamb, A.; Yamamoto, K.; and Ha, D. 2018. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*.
- Deng, A.; and Hooi, B. 2021. Graph Neural Network-Based Anomaly Detection in Multivariate Time Series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4027–4035.
- Deng, L. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6): 141–142.
- Dennis, D. K.; Li, T.; and Smith, V. 2021. Heterogeneity for the win: One-shot federated clustering. In *International conference on machine learning*, 2611–2620. PMLR.
- Garg, A.; Zhang, W.; Samaran, J.; Savitha, R.; and Foo, C.-S. 2021. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6): 2508–2517.
- Garst, S.; and Reinders, M. 2024. Federated k-means clustering. In *International Conference on Pattern Recognition*, 107–122. Springer.
- Guo, X.; Gao, L.; Liu, X.; and Yin, J. 2017. Improved Deep Embedded Clustering with Local Structure Preservation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 1753–1759.
- Holzer, P.; Jacob, T.; and Kavane, S. 2023. Dynamically weighted federated k-means. *arXiv preprint arXiv:2310.14858*.
- Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.
- Jain, A. K. 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters*, 31(8): 651–666.
- Lewis, D. D.; Yang, Y.; Rose, T. G.; and Li, F. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr): 361–397.
- Lim, G. 2024. Data Imbalance Solutions Using Pseudo-Cluster and Two-Step Classification Methods. In *2024 IEEE International Conference on Big Data (BigData)*, 8751–8753. IEEE.
- Liu, F.; Zeng, C.; Zhang, L.; Zhou, Y.; Mu, Q.; Zhang, Y.; Zhang, L.; and Zhu, C. 2022. Fedtadbench: Federated time-series anomaly detection benchmark. In *2022 IEEE 24th int conf on high performance computing & communications; 8th int conf on data science & systems; 20th int conf on smart city; 8th int conf on dependability in sensor, cloud & big data systems & application (HPCC/DSS/smartCity/dependSys)*, 303–310. IEEE.
- Liu, J.; Yan, J.; and Zhang, Z.-Y. 2024. Ccfc: Bridging federated clustering and contrastive learning. *arXiv preprint arXiv:2401.06634*.
- Madiraju, N. S. 2018. *Deep temporal clustering: Fully unsupervised learning of time-domain features*. Master's thesis, Arizona State University.
- Malhotra, P.; Ramakrishnan, A.; Anand, G.; Vig, L.; Agarwal, P.; and Shroff, G. 2016. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. In *ICML 2016 Anomaly Detection Workshop*.
- Mashhadi, A.; Sterner, J.; and Murray, J. 2021. Deep Embedded Clustering of Urban Communities Using Federated Learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8.
- McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. Pmlr.
- Miklautz, L.; Klein, T.; Sidak, K.; Leiber, C.; Lang, T.; Shkabrui, A.; Tschitschek, S.; and Plant, C. 2025. Breaking the reclustering barrier in centroid-based deep clustering. In *International Conference on Learning Representations*, volume 2025, 100295–100335.
- Nguyen, D. C.; Ding, M.; Pathirana, P. N.; Seneviratne, A.; Li, J.; and Poor, H. V. 2021. Federated learning for internet of things: A comprehensive survey. *IEEE communications surveys & tutorials*, 23(3): 1622–1658.
- Obied, M. A.; Ghaleb, F. F.; Hassanien, A. E.; Abdelfattah, A. M.; and Zakaria, W. 2023. Deep clustering-based anomaly detection and health monitoring for satellite telemetry. *Big Data and Cognitive Computing*, 7(1): 39.
- Otsu, N. 1979. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1): 62–66.
- Pan, C.; Sima, J.; Prakash, S.; Rana, V.; and Milenkovic, O. 2023. Machine Unlearning of Federated Clusters. In *The Eleventh International Conference on Learning Representations*.
- Park, H.; Klabjan, D.; Braun, M. W.; Li, X.; and Ananthanarayanan, B. 2026. Class-Grouped Normalized Momentum and Faster Hyperparameter Exploration to Tackle Class Imbalance in Federated Learning. In *Forty-third International Conference on Machine Learning*.

Pfeifer, B.; Sirocchi, C.; Bloice, M. D.; Kreuzthaler, M.; and Urschler, M. 2024. Federated unsupervised random forest for privacy-preserving patient stratification. *Bioinformatics*, 40(Supplement_2): ii198–ii207.

Presotto, R.; Civitarese, G.; and Bettini, C. 2022. Fedclar: Federated clustering for personalized sensor-based human activity recognition. In *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 227–236. IEEE.

Rieke, N.; Hancox, J.; Li, W.; Milletari, F.; Roth, H. R.; Albarqouni, S.; Bakas, S.; Galtier, M. N.; Landman, B. A.; Maier-Hein, K.; et al. 2020. The future of digital health with federated learning. *NPJ Digital Medicine*, 3(1): 119.

Ruff, L.; Vandermeulen, R. A.; Goernitz, N.; Deecke, L.; Siddiqui, S. A.; Binder, A.; Müller, E.; and Kloft, M. 2018. Deep One-Class Classification. In *Proceedings of the 35th International Conference on Machine Learning*, 4393–4402. PMLR.

Scott, J.; Lampert, C. H.; and Saulpic, D. 2025. Differentially Private Federated k-Means Clustering with Server-Side Data. In *Forty-second International Conference on Machine Learning*.

Seol, M.; and Kim, T. 2023. Performance enhancement in federated learning by reducing class imbalance of non-iid data. *Sensors*, 23(3): 1152.

Stallmann, M.; and Wilbik, A. 2022. Towards federated clustering: A federated fuzzy c-Means algorithm (FFCM). *arXiv preprint arXiv:2201.07316*.

Su, Y.; Zhao, Y.; Niu, C.; Liu, R.; Sun, W.; and Pei, D. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Suayama, K.; and Nakamura, K. 2025. Reducing Implicit Class Imbalance in Unlabeled Datasets Using Text-Specified Sensitive Attributes. In *2025 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2032–2037. IEEE.

Tuli, S.; Casale, G.; and Jennings, N. R. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proceedings of the VLDB Endowment*, 15(6): 1201–1214.

Wang, L.; Xu, S.; Wang, X.; and Zhu, Q. 2021. Addressing class imbalance in federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 10165–10173.

Wei, Y.; Peng, J.; He, T.; Xu, C.; Zhang, J.; Pan, S.; and Chen, S. 2023. Compatible transformer for irregularly sampled multivariate time series. In *2023 IEEE International Conference on Data Mining (ICDM)*, 1409–1414. IEEE.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487. PMLR.

Xu, X.; Liao, H.; and Yang, X. 2025. Federated deep embedded clustering under privacy protection. *Applied Soft Computing*, 185: 113963.

Yang, B.; Fu, X.; Sidiropoulos, N. D.; and Hong, M. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *International Conference on Machine Learning*, 3861–3870. PMLR.

Zong, B.; Song, Q.; Min, M. R.; Cheng, W.; Lumezanu, C.; Cho, D.; and Chen, H. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*.

A. Additional Method Details

A.1. k-FED Centroid Initialization and Restarting

Fed-BRDECS uses k-FED (Dennis, Li, and Smith 2021) for both initial centroid construction and periodic centroid restarting. We use k-FED because it constructs global centroids from uploaded local cluster means without requiring clients to transmit local soft assignment statistics or cluster-count statistics, which may reveal sensitive information about client-specific data composition.

After federated autoencoder pretraining, each client embeds its local data using the current encoder and solves a local clustering subproblem in the latent space. Following k-FED, the client sends only the resulting local cluster means to the server without sending cluster assignment statistics. The server then applies the k-FED server-side aggregation procedure to the collection of uploaded local means to obtain the global centroids. This procedure replaces the centralized k-means initialization used in DEC/IDEC with a one-shot federated centroid construction step.

The same procedure is used during restarting. At the beginning of each restart iteration, the server does not reset or perturb the encoder or decoder parameters as in BRB (Miklautz et al. 2025). Instead, clients recompute local cluster means from their current embeddings, and the server reruns the k-FED aggregation step to refresh the global centroids. The model parameters are kept unchanged across restarts, while the centroids are relocated according to the current federated embedding geometry. This design allows Fed-BRDECS to correct biased or inactive centroids induced by heterogeneous client updates without disrupting the learned representation.

Formally, at restart r , clients compute embeddings using the current encoder $f_{\theta^{r,0}}$ and the server updates

$$\{\mu_j^r\}_{j=1}^C \leftarrow \text{k-FED}(\{f_{\theta^{r,0}}(\mathcal{D}_k)\}_{k=1}^K).$$

The model parameters are then set to

$$\Theta^{r,0} = \{\theta^{r,0}, \phi^{r,0}, \mu_1^r, \dots, \mu_C^r\},$$

and federated refinement proceeds from this centroid-refreshed model.

A.2. Privacy Scope of Fed-BRDECS

Fed-BRDECS is privacy-preserving in the specific sense of reducing local assignment-statistic disclosure. In DEC/IDEC-style federated refinement, local soft-assignment matrices, soft-count vectors, or cluster-frequency statistics

directly encode client-specific cluster composition. Such composition information can be sensitive in practical applications, for example when local clusters correspond to defect types, disease subpopulations, or machine operating regimes.

Fed-BRDECS does not provide formal differential privacy. Like other prototype-based federated clustering methods, it still communicates model updates and local centroid means during k-FED initialization and restarting. However, during clustering refinement, Fed-BRDECS avoids repeated transmission of local soft assignments, soft-count vectors, and predicted-cluster histograms. Existing work (Scott, Lampert, and Saulpic 2025) applies differential privacy to mitigate this leakage, but DP-noised cluster counts may still reveal client-level composition. Appendix further discusses why adding DP noise to cluster counts may fail to hide local composition unless the noise magnitude is large enough to substantially degrade the utility of the statistics.

A.3. Why DP-Noised Cluster Counts Still Reveal Local Composition

A natural alternative to avoiding local assignment statistics is to transmit differentially private noisy cluster counts. We argue that this only partially addresses the privacy issue considered in this work. Differential privacy primarily protects individual-record participation, whereas our concern is client-level cluster-composition disclosure. In federated clustering, a local cluster-count vector directly encodes the client’s data composition, which can be sensitive in applications such as industrial defect monitoring, where defect-type proportions may reveal process quality, machine conditions, or yield-related information.

Consider client k with n_k samples and C clusters. Let

$$c_k = (c_{k,1}, \dots, c_{k,C}) \in \mathbb{N}^C, \quad \sum_{j=1}^C c_{k,j} = n_k,$$

denote its local hard cluster-count vector, and let

$$\pi_k = \frac{c_k}{n_k}$$

be the corresponding local cluster-composition vector. For DEC/IDEC-style federated refinement, the analogous soft-count vector is

$$f_k = (f_{k,1}, \dots, f_{k,C}), \quad f_{k,j} = \sum_{i=1}^{n_k} q_{kij},$$

whose normalized version similarly reveals the client’s soft cluster composition.

Suppose a client applies the Laplace mechanism to its count vector:

$$\tilde{c}_k = c_k + \eta, \quad \eta_j \sim \text{Laplace}\left(\frac{\Delta}{\varepsilon}\right),$$

where Δ is the ℓ_1 -sensitivity of the histogram query. The server can estimate the client’s local composition by normalizing the noisy counts:

$$\tilde{\pi}_k = \frac{\tilde{c}_k}{n_k}.$$

Then

$$\mathbb{E}\|\tilde{\pi}_k - \pi_k\|_1 = \frac{1}{n_k} \mathbb{E}\|\eta\|_1 = \frac{C\Delta}{n_k\varepsilon}.$$

Therefore, for moderate or large local datasets, the normalized noise can be very small unless ε is chosen extremely small. For example, when $C = 10$, $n_k = 5,000$, $\Delta = 1$, and $\varepsilon = 1$, the expected ℓ_1 error in the recovered composition is only

$$\frac{10}{5000} = 0.002.$$

Thus, even after adding standard DP noise to the count vector, the server can still recover a highly accurate approximation of the client’s local cluster composition.

This issue can also be understood through group privacy. Differential privacy protects neighboring datasets that differ in one record, but two clients with different cluster compositions may differ in many records. Let π and π' be two possible local composition vectors. Transforming a histogram with composition π into one with composition π' requires

$$m = \frac{n_k}{2} \|\pi - \pi'\|_1$$

sample replacements. By group privacy, an ε -DP mechanism provides only $m\varepsilon$ -level indistinguishability for this composition-level change. Hence, to hide two compositions separated by $\delta = \|\pi - \pi'\|_1$, we need

$$\frac{n_k\varepsilon\delta}{2} = O(1), \quad \text{or equivalently} \quad \varepsilon = O\left(\frac{1}{n_k\delta}\right).$$

For a fixed composition gap δ , this requires $\varepsilon = O(1/n_k)$. Under the Laplace mechanism, the corresponding noise scale becomes

$$\frac{\Delta}{\varepsilon} = O(n_k),$$

meaning that the noise added to each count is on the same order as the client’s dataset size. Such noise would substantially corrupt the count vector and degrade its utility for federated k-means updates or DEC/IDEC target distribution construction.

Therefore, DP-noised cluster counts exhibit a sharp privacy–utility tradeoff for the composition-disclosure threat considered in this work. Moderate noise preserves clustering utility but still reveals an accurate estimate of local cluster composition; noise large enough to obscure that composition makes the statistics unreliable for clustering. Fed-BRDECS avoids this disclosure channel by not transmitting local soft-assignment matrices, local cluster-frequency vectors, or predicted-cluster histograms during refinement.

B. Experiment Details

B.1. Datasets and Models

Image datasets. For MNIST (Deng 2012), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), and KMNIST (Clanuwat et al. 2018), we use the official test split of each dataset. Each image is flattened into a vector before being fed into the autoencoder. The encoder follows the standard DEC-style multilayer perceptron architecture

$$784 \rightarrow 500 \rightarrow 500 \rightarrow 2000 \rightarrow 10,$$

where the final layer gives a 10-dimensional latent representation. The decoder is symmetric. The number of clusters is set to the number of ground-truth classes, i.e., $C = 10$. Labels are used only for partitioning and evaluation, not during training.

REUTERS-10K. For REUTERS-10K, we follow the pre-processing protocol of DEC (Xie, Girshick, and Farhadi 2016). We use four root categories (corporate/industrial, government/social, markets, and economics) as labels and exclude documents with multiple labels. Following DEC’s protocol, we also compute tf-idf features using the 2,000 most frequent words and use them as input. Due to computational constraints, we randomly sample 10,000 examples. We refer to this sampled dataset as REUTERS-10K.

For this dataset, the encoder architecture is

$$2000 \rightarrow 500 \rightarrow 500 \rightarrow 2000 \rightarrow 10,$$

with a symmetric decoder and a 10-dimensional latent representation.

B.2. Implementation Details

For centroid-based baselines, including k-FED, FKM, and DWF k-Means, we apply the algorithms directly to the raw input features, following the evaluation setting of the original federated clustering papers. Thus, for image datasets, these methods operate on flattened raw pixels, while for REUTERS-10K they operate on tf-idf features. Deep clustering methods instead learn an autoencoder representation before or during clustering refinement. All federated deep clustering methods use FedAvg as the federated optimization framework. For methods incorporating the sample stability loss or the balanced sampler, each client updates the sampling weights or calculates the Otsu threshold τ_k once per epoch across their entire local dataset.

Selection of loss weights. We examine the sensitivity of Fed-BRDECS to the clustering-loss weight λ_{clu} on MNIST and Fashion-MNIST. As shown in Figure 3 in the main body, the preferred weight differs between the IID and non-IID settings. We use $\lambda_{\text{clu}} = 1.0$ for IID partitions, which provides near-optimal performance on both datasets, and $\lambda_{\text{clu}} = 0.5$ for non-IID partitions, which achieves the highest ACC on both datasets. These values are used for the corresponding partition settings in all main experiments. We set $\lambda_{\text{var}} = 0.8$, following Cheng et al. (2024).

Optimization Details Local client optimization uses SGD with momentum. We use a cosine annealing learning-rate scheduler for both the pretraining and refinement stages. The scheduler decays the learning rate from the selected initial value to 10^{-3} times the initial value by the final round. For refinement with multiple restarts, the scheduler is applied over the entire refinement process rather than being reset at each restart. We tune weight decay over

$$\{10^{-4}, 10^{-5}, 0\},$$

and batch size over

$$\{64, 128, 256\}.$$

The initial learning rate is selected by grid search from

$$\{1.0, 0.2, 0.1, 0.02, 0.01\}.$$

For local training, each client performs 20 local optimization steps per communication round. All experiments are implemented in PyTorch 2.1.0 with CUDA 12.6. Jobs are executed on four NVIDIA GeForce RTX 2080 Ti GPUs.

B.3. Full Clustering Results

Table 3 reports the full numerical clustering results corresponding to Figure 2 in the main body. All ACC and NMI values are averaged over five independent runs with different random seeds.

C. Additional Details for Federated Time-Series Anomaly Detection

C.1. Related Work

Centralized TSAD spans deep one-class methods (Deep SVDD (Ruff et al. 2018)), reconstruction and forecasting methods (LSTM-AE (Malhotra et al. 2016), OmniAnomaly (Su et al. 2019), USAD (Audibert et al. 2020)), graph-based methods (GDN (Deng and Hooi 2021)), and transformer-based detectors (TranAD (Tuli, Casale, and Jennings 2022)). A growing thread also builds detectors on deep embedded clustering itself, including DTC (Madiraju 2018) and DCLOP (Obied et al. 2023). Such DEC-based detectors are common in centralized learning but scarce in federated learning, where FedTADBench (Liu et al. 2022) evaluates representative TSAD algorithms under federated learning. Fed-BRDECS fills this gap as a federated deep clustering backbone for unsupervised outlier detection, whose mechanism can be easily plugged into existing federated TSAD algorithms.

C.2. Implementation Details

General strategy to instantiate Fed-BRDECS. Let $x_{k,i} \in \mathbb{R}^{W \times d}$ denote a time-series window from client k , where W is the window length and d is the dimensionality. Let $\mathcal{B}_k \subset \mathcal{D}_k$ denote a mini-batch of local windows sampled on client k . A reconstruction-based TSAD backbone maps $x_{k,i}$ to a latent representation $h_{k,i}$ and reconstructs the input as $\hat{x}_{k,i}$. Its native training objective is the reconstruction loss

$$\mathcal{L}_{\text{rec}}^{(k)} = \frac{1}{|\mathcal{B}_k|} \sum_{x_{k,i} \in \mathcal{B}_k} \ell_{\text{rec}}(x_{k,i}, \hat{x}_{k,i}),$$

where ℓ_{rec} denotes the backbone-specific reconstruction loss. In our experiments, we use the mean squared reconstruction error over the window,

$$\ell_{\text{rec}}(x_{k,i}, \hat{x}_{k,i}) = \frac{1}{Wd} \|x_{k,i} - \hat{x}_{k,i}\|_F^2.$$

At inference time, the outlier score is computed from the same reconstruction error.

To instantiate Fed-BRDECS, we attach a lightweight projection head p_ω to the backbone latent representation

$$z_{k,i} = p_\omega(h_{k,i}) \in \mathbb{R}^{d_c},$$

Table 3: Full clustering results on IID and non-IID federated partitions with $K = 10$ clients. Each entry reports ACC and NMI in parentheses. The non-IID setting uses a Dirichlet partition with $\alpha = 0.5$. All values are averaged over five independent runs with different random seeds. Higher is better. Best ACC results are shown in bold, and best NMI results are bolded in parentheses. Lower performance of DEC-based refinement compared with Fed-AE indicates that the refinement stage can degrade the pretrained representation, especially under non-IID client distributions.

Dataset / Partition	k-FED	FKM	DWF	CCFC	Fed-AE	F-DEC	Fed-DECS	Fed-BDECS	Fed-BRDECS	Fed-BRIDECS
MNIST / IID	0.4530 (0.4990)	0.4866 (0.4347)	0.4914 (0.4318)	0.8576 (0.8345)	0.8155 (0.7531)	0.8878 (0.9259)	0.8394 (0.7853)	0.8408 (0.7862)	0.9210 (0.8782)	0.9010 (0.8979)
MNIST / Non-IID	0.5524 (0.4896)	0.5130 (0.4563)	0.4972 (0.4338)	0.8026 (0.7761)	0.7973 (0.7296)	0.7016 (0.8062)	0.8215 (0.7638)	0.8215 (0.7703)	0.9225 (0.8498)	0.7158 (0.5969)
Fashion-MNIST / IID	0.5297 (0.5228)	0.4964 (0.5091)	0.4745 (0.5170)	0.5448 (0.5798)	0.5409 (0.5618)	0.5538 (0.5352)	0.5438 (0.5748)	0.5477 (0.5732)	0.5832 (0.5889)	0.5576 (0.5865)
Fashion-MNIST / Non-IID	0.4976 (0.4959)	0.4987 (0.5322)	0.5446 (0.5068)	0.5231 (0.5683)	0.5614 (0.5797)	0.5024 (0.5670)	0.5508 (0.5824)	0.5685 (0.5764)	0.6129 (0.5925)	0.5438 (0.5805)
KMNIST / IID	0.3066 (0.2790)	0.3798 (0.2667)	0.3421 (0.2521)	0.4685 (0.4345)	0.4642 (0.4294)	0.4722 (0.4503)	0.4735 (0.4267)	0.4745 (0.4328)	0.4798 (0.4381)	0.4735 (0.4405)
KMNIST / Non-IID	0.3100 (0.2574)	0.3059 (0.2709)	0.3352 (0.2656)	0.3954 (0.3725)	0.4099 (0.3678)	0.3854 (0.3932)	0.3765 (0.3724)	0.4122 (0.3745)	0.4338 (0.4065)	0.3978 (0.3899)
REUTERS-10K / IID	0.6873 (0.3945)	0.7590 (0.5098)	0.7600 (0.4400)	0.7716 (0.5630)	0.7285 (0.4904)	0.7606 (0.5580)	0.7536 (0.5387)	0.7559 (0.5742)	0.8039 (0.5749)	0.7832 (0.5439)
REUTERS-10K / Non-IID	0.5962 (0.3316)	0.7064 (0.4235)	0.6553 (0.3564)	0.7528 (0.5274)	0.7613 (0.5137)	0.7622 (0.5248)	0.7658 (0.4984)	0.7805 (0.4841)	0.7865 (0.5316)	0.7654 (0.4952)

where d_c is the clustering embedding dimension. Fed-BRDECS then maintains centroids $\{\mu_j\}_{j=1}^C$ in this clustering space. During training, soft assignments q_{kij} are computed from $z_{k,i}$ and the centroids using the same Student- t kernel as in the main method. The local objective becomes

$$\mathcal{L}_{\text{TSAD}}^{(k)} = \mathcal{L}_{\text{rec}}^{(k)} + \lambda_{\text{stab}} \mathcal{L}_{\text{stab}}^{(k)}.$$

Centroids are initialized and refreshed by k-FED on the client clustering embeddings. The prediction-balanced sampler is constructed from the induced pseudo-clusters but with smoothing parameterized by γ . Thus, windows assigned to locally rare predicted regimes are sampled more often, while the smoothing parameter γ controls the strength of rebalancing and prevents the sampler from over-concentrating on a small number of windows.

At inference time, the projection head and centroids are discarded, and outlier scores are computed only from the original reconstruction error. Therefore, Fed-BRDECS acts as a training-time representation-learning module and does not add inference-time cost.

LSTM-AE instantiation. For LSTM-AE, the encoder maps each input window $x_{k,i}$ to a final recurrent hidden state $h_{k,i}$. We attach the clustering projection head to this final hidden state:

$$z_{k,i} = p_{\omega}(h_{k,i}).$$

The decoder reconstructs the input window, and the reconstruction loss is the mean squared error between $x_{k,i}$ and $\hat{x}_{k,i}$. Thus, the LSTM-AE backbone is unchanged except for the temporary clustering head used during training. Since $h_{k,i}$ is already a single vector representation of the full window, no temporal pooling is required before clustering.

TranAD instantiation. TranAD uses a two-phase transformer reconstruction mechanism. Given an input window, the first decoder produces an initial reconstruction, and the second decoder reconstructs the target using the first-phase reconstruction error as additional context. We attach the clustering head to the latent representation produced by the first-phase transformer encoder. Let

$$M_{k,i}^{(1)} \in \mathbb{R}^{W \times d_h}$$

denote the first-phase encoder memory for window $x_{k,i}$. We obtain a window-level representation by mean pooling over

time:

$$h_{k,i} = \frac{1}{W} \sum_{t=1}^W M_{k,i,t}^{(1)}.$$

The clustering embedding is then

$$z_{k,i} = p_{\omega}(h_{k,i}).$$

The native TranAD reconstruction loss is kept unchanged. In particular, if $x_{k,i}^{(1)}$ and $x_{k,i}^{(2)}$ denote the first- and second-phase reconstructions, the local reconstruction loss uses the same epoch-dependent weighting as TranAD:

$$\ell_{\text{rec}} = \frac{1}{e} \ell(x_{k,i}^{(1)}, x_{k,i}) + \left(1 - \frac{1}{e}\right) \ell(x_{k,i}^{(2)}, x_{k,i}),$$

where e is the local epoch index. At inference time, the outlier score is computed from the second-phase reconstruction error, as in the original TranAD.

Prediction-balanced sampling for TSAD. For each client, Fed-BRDECS periodically constructs a prediction-balanced sampler from the pseudo-clusters induced by the temporary clustering head. The purpose is to increase the exposure of locally rare normal regimes during training, while avoiding overly aggressive oversampling that may overfit a small number of anomalous windows.

Specifically, client k computes soft assignments over its local windows and estimates the local soft cluster mass

$$f_{k,j} = \sum_{x_{k,i} \in \mathcal{D}_k} q_{kij}, \quad j = 1, \dots, C.$$

Each window is assigned to its most likely pseudo-cluster,

$$\hat{y}_{k,i} = \arg \max_{j \in \{1, \dots, C\}} q_{kij}.$$

The inverse-frequency cluster weight is then smoothed by an exponent γ :

$$\tilde{w}_{k,j} = \left(\frac{1}{f_{k,j} + \epsilon} \right)^{\gamma},$$

and each window receives the sampling weight

$$w_{k,i} = \tilde{w}_{k,\hat{y}_{k,i}}.$$

Mini-batches are sampled with replacement according to probabilities proportional to $w_{k,i}$. Here, γ controls the strength of rebalancing: $\gamma = 0$ recovers uniform sampling,

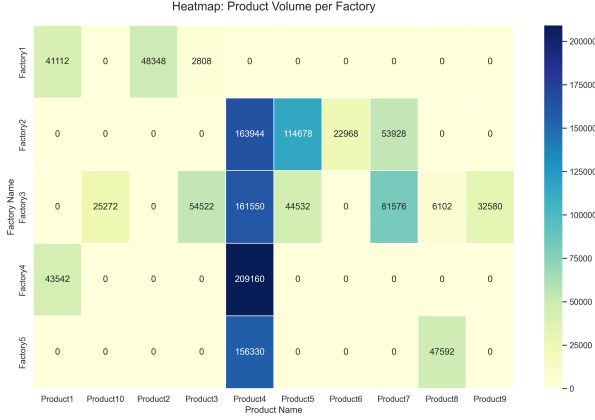


Figure 4: Factory–product production-volume metadata for the proprietary sensor dataset. The heatmap shows strong non-IID structure across factories.

while $\gamma = 1$ corresponds to the standard inverse-frequency sampler. This smoothing is used to prevent the sampler from concentrating too heavily on rare pseudo-clusters, which may otherwise increase the risk of overfitting to anomalous or atypical windows. In our TSAD experiments, we use $\gamma = 0.5$.

Centroid initialization and restarting. After reconstruction pretraining, each client extracts clustering embeddings $\{z_{k,i}\}$ from its local windows and runs local k-means. The server aggregates the returned local centroids using k-FED to obtain the initial global centroids. During refinement, Fed-BRDECS periodically refreshes centroids by rerunning the same k-FED procedure on the current clustering embeddings. This centroid-level restarting corrects biased or inactive centroids caused by heterogeneous local time-series regimes. No clustering head is used at inference.

Details for proprietary sensor dataset. The proprietary sensor dataset is collected from industrial manufacturing equipment and consists of irregularly sampled multivariate time-series samples. After preprocessing, the dataset contains 72,808 samples with an anomaly rate of 3.12%. The anomaly labels are used only for evaluation and are not used during unsupervised federated training.

The dataset is naturally federated by factory, where each factory is treated as one client. Figure 4 summarizes the factory–product metadata. The metadata covers five factories and ten products, but the support is sparse, indicating the strong client heterogeneity.

Figure 5 shows representative multivariate trajectories from three product types. The trajectories exhibit irregular temporal structure, product-dependent sequence lengths, sharp transitions, and different sensor-channel dynamics. These observations support the use of a model designed for irregularly sampled time series.

CoFormer-AE for irregularly sampled sensor data. The proprietary sensor dataset consists of irregularly sampled multivariate time series. Therefore, we use CoFormer-AE instead of LSTM-AE or TranAD for this dataset. CoFormer

provides the window-level representation $h_{k,i}$ used by the reconstruction model, and Fed-BRDECS is attached in the same way

$$z_{k,i} = p_{\omega}(h_{k,i}).$$

The remaining procedure, including k-FED centroid initialization, sample-stability loss, prediction-balanced sampling, and inference by reconstruction error, is unchanged. We do not evaluate TranAD on this dataset because TranAD is architecture-specific to regularly sampled fixed-window inputs and is not directly compatible with the CoFormer backbone.

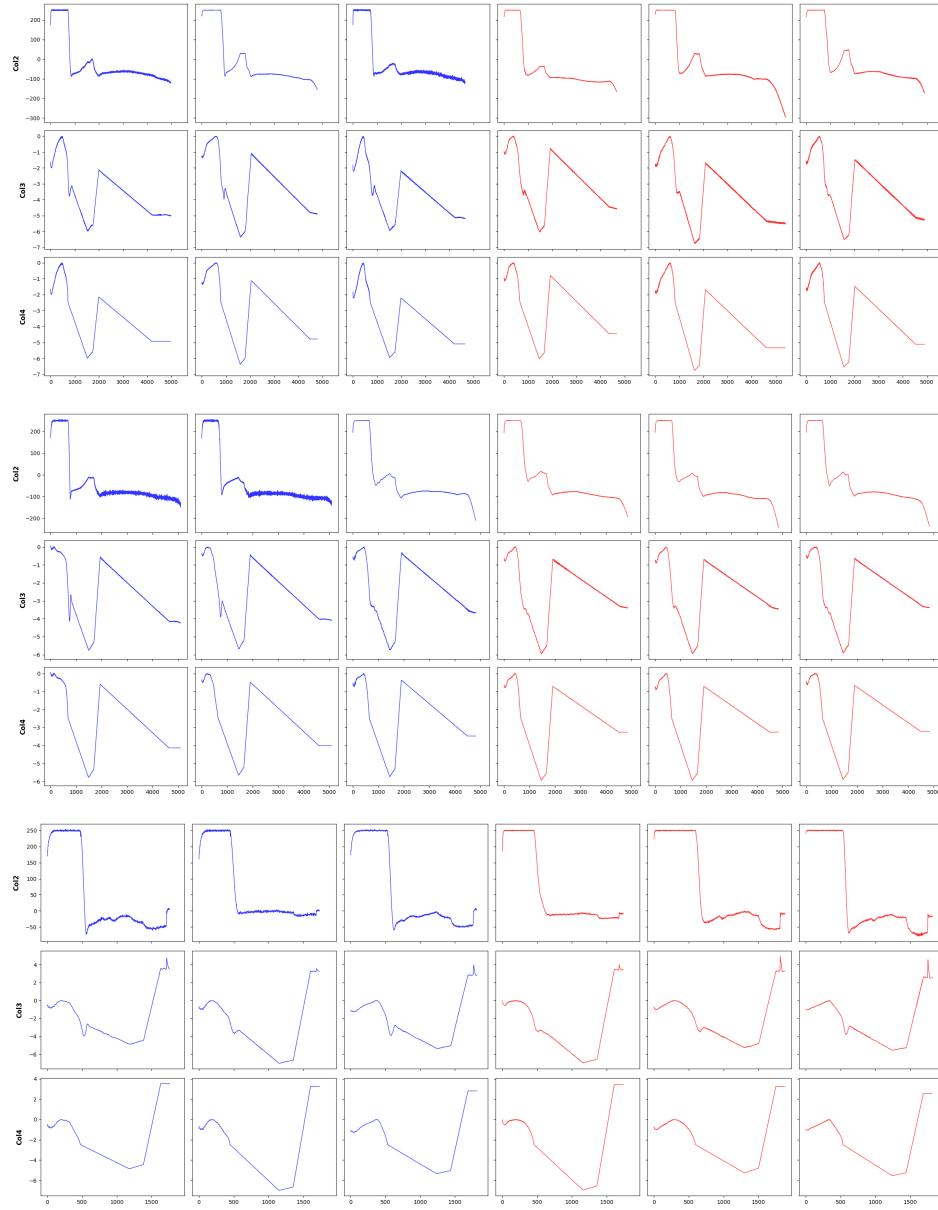


Figure 5: Representative multivariate sensor trajectories from Products 1, 2, and 4 in the proprietary dataset. Each panel visualizes multiple examples over three sensor channels. The examples show product-dependent temporal structure, variable sequence length, sharp transitions, plateau regions, and heterogeneous channel dynamics, motivating the use of CoFormer-AE for irregularly sampled sensor data.