FED-DREAM: BRIDGING FRAGMENTED VIEWS IN HYBRID FEDERATED LEARNING VIA RELATIONAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Hybrid federated learning (HBFL) addresses scenarios in which sample and feature spaces are fragmented across clients, so that each client holds only a partial view with limited predictive capacity. Existing HBFL methods aggregate views through feature or prediction sharing with static fusion strategies, missing the inter-client relations essential for a holistic representation. We propose **Fed-DREAM** (Federated Divisive RElational Aggregation and Mixup), a framework for collaborative relational learning in HBFL built on a three-stage pipeline: (1) *federated divisive clustering*, which constructs structurally aligned global clusters without exposing raw data, (2) *multi-sample representation-level mixup* for privacy-preserving synthetic feature sharing, and (3) an *edge-conditioned gated graph neural network* that fuses fragmented views by conditioning message passing on inter-client feature-membership relations. Experiments across tabular, image, and text modalities show that Fed-DREAM consistently outperforms state-of-the-art benchmarks while exhibiting lower membership-inference leakage than the highest-utility baseline.

1 INTRODUCTION

Traditional cloud-based machine learning applications operating in multi-client environments often assume that datasets from diverse clients can be centrally consolidated and processed to maximize prediction performance. However, such centralization is frequently infeasible or undesirable due to strict data privacy regulations, prohibitive communication costs, and inherent data heterogeneity (Zhang et al., 2020b); federated learning (FL) instead enables both homogeneous and heterogeneous clients to collaboratively train models without exposing their raw data (Kairouz et al., 2021).

FL is generally categorized into horizontal FL (HFL), vertical FL (VFL), and hybrid FL (HBFL) by the source of data heterogeneity (Zhang et al., 2020b). HFL and VFL address sample and feature heterogeneity, respectively, whereas HBFL handles scenarios in which both are fragmented across clients: with global features $\mathcal{F} = \bigcup_{m=1}^{M} \mathcal{F}_m$, each client m observes only a subset $\mathcal{F}_m$ together with a subset of samples, yielding partial views that reflect real-world constraints in domains such as medicine and finance (Ng et al., 2021; Gao et al., 2023). Since neither a single client nor HFL/VFL alone can reconstruct a holistic representation from such views, HBFL requires dedicated collaborative mechanisms; yet research on it remains scarce, and the few deep learning methods proposed fall into two strategies: constructing a shared global or auxiliary model via parameter aggregation, as in HFL (Zhang et al., 2020b; Gao et al., 2023), or synchronizing labels to align targets across clients (Llasag Rosero et al., 2024; 2025).

However, existing HBFL methods focus on static model alignment, which relies on fixed, sample-agnostic aggregation weights or simple prediction sharing. These rigid dependencies discard the relational context essential for reconciling fragmented views, thus limiting predictive utility. Furthermore, existing HBFL frameworks often diverge in their primary objectives. Some prioritize local model enhancement but overlook the architectural requirements for secure global inference. Others attempt to facilitate global intelligence through the explicit sharing of predictions or sensitive representations, which raises significant privacy concerns.

To address this, we propose **Fed-DREAM** (Federated Divisive RElational Aggregation and Mixup), a framework for collaborative relational learning that fuses client representations at the server to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

reconstruct a holistic view of each sample under privacy-preserving training and joint inference. It comprises three stages: (1) *federated divisive clustering*, which securely builds structurally aligned global clusters as topological templates for collaboration; (2) *multi-sample representation-level mixup*, which releases Dirichlet-interpolated synthetic representations that protect privacy and augment training; and (3) an *edge-conditioned gated graph neural network* (ECG-GNN), which fuses these representations by conditioning both message generation and gating on inter-client feature relationships, integrating complementary information while handling missing observations.

Our main contributions are summarized as follows:

- A collaborative relational learning framework for HBFL in which ECG-GNN, a relation-aware aggregator, lets clients jointly construct holistic representations from fragmented views while handling missing observations.
- A privacy-preserving representation sharing mechanism combining federated divisive clustering and multi-sample representation-level mixup, which enhances data confidentiality and predictive robustness.
- Consistent gains over state-of-the-art benchmarks across tabular, image, and text modalities, together with an empirical membership-inference audit showing lower leakage than the highest-utility baseline.

2 Related Works

HFL methods (McMahan et al., 2017; Li et al., 2020; Karimireddy et al., 2020; Acar et al., 2021) assume a shared feature space and VFL (Hardy et al., 2017; Liu et al., 2021; Castiglia et al., 2022; Khan et al., 2022; He et al., 2025) relies on sample alignment with full participation, so neither applies when both sample and feature spaces are fragmented; HBFL has emerged to enable collaborative training under these dual heterogeneities.

Early HBFL research used specialized optimization (Overman et al., 2024) or secure sample alignment (Chen et al., 2021; Cheng et al., 2021), and federated transfer learning bridged heterogeneous feature spaces via cross-domain adaptation (Gao et al., 2019; Sharma et al., 2019); even with deep learning extensions (Liu et al., 2020; Kang et al., 2022), these remain restricted to dual-party or structurally constrained settings and do not scale to high-dimensional, multi-client HBFL.

To handle high-dimensional HBFL across numerous clients, recent works have introduced specialized deep learning mechanisms. HyFEM (Zhang et al., 2020b) utilized model partitioning and matching mechanisms to assimilate diverse local feature extractors into a unified server model, which is then personalized for individual clients. To mitigate the client drift arising from incomplete local views, FedHD (Gao et al., 2023) employed tracking variables to incorporate global gradient information into local updates. Furthermore, FedLabSync (Llasag Rosero et al., 2024; 2025) addressed severe label fragmentation and structural heterogeneity by synchronizing knowledge through penalized collaborative optimization, leveraging predictions from the other clients to align disparate models. Recent efforts have also explored mixed horizontal–vertical partitions in applied domains such as financial crime detection (Khan et al., 2026); these likewise rely on static fusion without modeling inter-client relations.

Despite this progress, current HBFL frameworks synchronize decentralized models through static parameter aggregation or rigid prediction alignment, leaving structural interactions among clients unaddressed, and the predictions or representations they exchange during training are shared without obfuscation. Fed-DREAM addresses both by learning holistic representations relationally while releasing only obfuscated training representations, keeping raw data and individual local representations on the clients.

3 Method

3.1 Problem Formulation

Consider a global training dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$ of N samples with complete feature vectors $\mathbf{x}_n$ and labels $\mathbf{y}_n$. In an HBFL setting with M clients, let $\mathcal{N}_m$ and $\mathcal{F}_m$ denote the sample and feature index sets available to client $m \in \{1, \ldots, M\}$. Client m privately holds $\mathcal{D}_m = \{(\mathbf{x}_{m,n}, \mathbf{y}_n) \mid n \in$

$\mathcal{N}_m\}$, where $\mathbf{x}_{m,n}$ contains only the attributes indexed by $\mathcal{F}_m$, along with the label $\mathbf{y}_n$ of every sample $n \in \mathcal{N}_m$ it holds, as in cross-silo settings where each institution records the outcome of its own patients or customers; label fragmentation across clients (Llasag Rosero et al., 2024; 2025) is outside our scope. Since clients hold their respective attributes for partially overlapping instance IDs, the server is assumed to know the index allocations $(\mathcal{N}_m, \mathcal{F}_m)$ but never the raw feature values. Each client m maintains a local parameter set $\theta_m = \{\theta_m^{emb}, \theta_m^{head}\}$ parameterizing a local embedding model f_m^{emb} and a prediction head f_m^{head}.

We formulate the cooperative training process as a joint optimization over client-side and server-side parameters. While the server remains oblivious to the raw private data $\mathcal{D}_m$, it relies solely on the outputs of two client-side black-box mappings: a local update mapping $\Omega_m(\cdot)$ and a privacy operator $\Phi_m(\cdot)$, both operating over the global consensus parameters Θ, which are partitioned according to the sharing configurations of Section 3.2.

The local update mapping yields the shareable local parameters $\theta_m^{sh} = \Omega_m(\Theta) \subseteq \theta_m$, $\forall m \in \{1, \ldots, M\}$, which client m locally optimizes on $\mathcal{D}_m$. The privacy operator $\Phi_m(\cdot)$ then acts on the updated θ_m^{sh} to generate an obfuscated local synthetic dataset

$$\tilde{\mathcal{D}}_m = \Phi_m(\theta_m^{sh}) = \{(\tilde{\mathbf{e}}_m^{(i)}, \tilde{\mathbf{y}}_m^{(i)})\}_{i \in \mathcal{I}_m}, \tag{1}$$

where $\mathcal{I}_m$ indexes the synthetic releases of client m, and $\Phi_m(\theta_m^{sh})_X$ and $\Phi_m(\theta_m^{sh})_Y$ denote the synthetic representations and targets, respectively. $\Phi_m(\cdot)$ is designed as a one-way mapping with respect to $\mathcal{D}_m$.

Composing the two mappings yields the overall objective over Θ and the server-side parameters θ_g:

$$\min_{\theta_g, \Theta} \mathcal{L}\Big(\Upsilon\big(\{\Phi_m(\Omega_m(\Theta))_X\}_{m=1}^M; \theta_g\big),\ \{\Phi_m(\Omega_m(\Theta))_Y\}_{m=1}^M\Big), \tag{2}$$

where $\Upsilon(\cdot; \theta_g)$ is a global prediction function producing a single prediction from the collection of client outputs, and $\mathcal{L}$ is a set-level loss between the predictions and the collection of local synthetic targets.

Since (2) composes black-box mappings, we minimize it in phases rather than in one shot. Clients first optimize θ_m^{sh} in parallel and the server synchronizes the consensus parameters, $\Theta = \text{Aggregate}(\{\theta_m^{sh}\}_{m=1}^M)$, following the protocol of Section 3.2; the privacy operator $\Phi_m(\cdot)$ is then realized by federated divisive clustering (Section 3.3) followed by multi-sample representation-level mixup (Section 3.4); finally, $\Upsilon(\cdot; \theta_g)$ is instantiated as an ECG-GNN at the server and trained on $\{\tilde{\mathcal{D}}_m\}_{m=1}^M$ (Section 3.5).

3.2 Local Model Optimization

Throughout local update cycles, clients can potentially synchronize their network parameters to benefit from collaborative learning. However, in HBFL, this parameter sharing strategy must be flexibly determined based on architectural compatibility, privacy regulations, and the risk of negative transfer among heterogeneous clients. We categorize these configurations into three cases. Under *no parameter sharing*, models are trained strictly locally, which is adopted when modules are structurally incompatible or when clients prioritize maximum isolation. Under *selective parameter sharing*, only specific modules (e.g., encoders over overlapping features) are collaboratively learned while the rest remain private, balancing collaboration against privacy constraints (Zhang et al., 2020b). Under *global parameter sharing*, all shareable parameters are synchronized via federated optimization (e.g., FedAvg (McMahan et al., 2017)) when inputs are compatible and all clients consent.

To accommodate these diverse and discretionary configurations, we generalize our local training process as localized training with conditional communication. Let $\mathcal{K}_m$ denote the index set of shareable modules assigned to client m. The global consensus parameters Θ maintained at the server are structured as a collection of distinct modules, denoted as $\{\Theta^{(k)}\}_{k \in \bigcup_{m=1}^M \mathcal{K}_m}$, where $\Theta^{(k)}$ represents the k-th global shareable module.

We strictly partition the complete parameter set of client m, denoted as $\theta_m = \theta_m^{emb} \cup \theta_m^{head}$, into a shareable parameter subset θ_m^{sh} and a private parameter subset θ_m^{pr}. Here, the shareable subset directly aggregates the specific blocks governed by the client's configuration: $\theta_m^{sh} = \{\Theta^{(k)}\}_{k \in \mathcal{K}_m}$.

If all clients opt out of sharing ($\mathcal{K}_m = \emptyset, \forall m$), the framework naturally defaults to standalone local training.

By tying the local shared modules to the global consensus parameters Θ, the overarching learning objective for the local optimization phase is formulated as a joint federated optimization problem:

$$\min_{\Theta, \{\theta_m^{pr}\}_{m=1}^M} \sum_{m=1}^{M} \frac{|\mathcal{D}_m|}{\sum_{j=1}^{M} |\mathcal{D}_j|} \left(\frac{1}{|\mathcal{D}_m|} \sum_{n \in \mathcal{N}_m} \ell\Big(\mathbf{y}_n, \sigma\big(f_m^{head}\big(f_m^{emb}(\mathbf{x}_{m,n})\big)\big)\Big) \right), \tag{3}$$

where $\sigma(\cdot)$ denotes the activation function depending on the prediction task.

The detailed procedure for this generalized local optimization is summarized in Appendix A.1. Furthermore, we introduce an adaptive protocol that determines the optimal encoder depth (L) from a pre-defined local architecture, the details of which are provided in Appendix A.2.

3.3 Privacy-Preserving Federated Divisive Clustering

Following local optimization, our framework partitions the N samples into structurally aligned global clusters $\mathcal{G} = \{G_1, \ldots, G_K\}$, $|G_k| \in [C, 2C-1]$, in the latent space of the local encoders $\{f_m^{emb}\}_{m=1}^M$ without accessing raw data; C guarantees enough samples for the subsequent mixup phase, and its selection is detailed in Appendix A.5. To secure the protocol against an honest-but-curious server, all distance computations use a threshold homomorphic encryption (THE) scheme (Paillier, 1999; Fouque et al., 2000; Damgård & Jurik, 2001) whose decryption key is secret-shared among clients, so the server obtains only aggregated results through collaborative decryption. The protocol proceeds in four stages, outlined below; the complete step-by-step procedure, including the batched on-demand distance retrieval and the size-constrained clustering subroutine, is given in Appendix A.3 together with the full pseudo-code (Algorithm 2).

Stage 1 (Secure distance aggregation, one-time and offline). For batches of sample-index pairs, each client encrypts its local pairwise distances, scaled to $[0, 1]$ by their maximum, and uploads them; the server homomorphically sums the ciphertexts, decrypts collaboratively, averages over the clients observing each pair, and returns each averaged batch to those clients.

Stage 2 (Local clustering). Each client assembles the returned distances over its own samples, runs a size-constrained divisive clustering (Algorithm 3), and uploads only cluster medoid indices and membership lists.

Stage 3 (Global medoid clustering). The server pools the medoids, merges memberships that share a medoid, and applies Algorithm 3 to the medoids themselves, using the average cross-cluster distance between member sets—retrieved on demand from their holders—as the inter-medoid distance.

Stage 4 (Refinement). The server aggregates the samples under each medoid cluster and recursively re-partitions any group of size $\geq 2C$, so every final cluster satisfies $|G_k| \in [C, 2C-1]$.

3.4 Multi-sample Representation-level Mixup

To further bolster data privacy and mitigate potential performance degradation caused by clustering, we introduce multi-sample representation-level mixup. This phase is designed as a federated protocol that accounts for partial sample availability across heterogeneous clients while ensuring that the server never accesses individual local embeddings or raw predictions during training. Unlike InstaHide (Huang et al., 2020), which mixes raw images and was later inverted by regressing the images from many mixtures (Carlini et al., 2021), we mix the outputs of a non-injective encoder, so that even a recovered embedding does not pin down the raw features (Appendix A.4).

For each global cluster G_k, each client m first identifies its local sample indices $\mathcal{N}_{m,k} = \mathcal{N}_m \cap G_k$, and then independently samples $|G_k|$ weight vectors $\boldsymbol{\lambda}^{(i)} \in \mathbb{R}^{|\mathcal{N}_{m,k}|}$, $i \in \{1, \ldots, |G_k|\}$, from $\mathrm{Dir}(\alpha)$ over $\mathcal{N}_{m,k}$; these weights are never disclosed to the server or to other clients. Using them, client m generates $|G_k|$ local synthetic embeddings $\tilde{\mathbf{e}}_{m,k}^{(i)}$ and logits $\tilde{\mathbf{l}}_{m,k}^{(i)}$:

$$\tilde{\mathbf{e}}_{m,k}^{(i)} = \sum_{n \in \mathcal{N}_{m,k}} \lambda_n^{(i)} f_m^{emb}(\mathbf{x}_{m,n}), \quad \tilde{\mathbf{l}}_{m,k}^{(i)} = \sum_{n \in \mathcal{N}_{m,k}} \lambda_n^{(i)} f_m^{head}(f_m^{emb}(\mathbf{x}_{m,n})). \tag{4}$$

Clients with $\mathcal{N}_{m,k} = \emptyset$ are excluded from this process. When $|\mathcal{N}_{m,k}| = 1$, client m's releases for G_k reduce to its raw embedding; this corresponds to the $C = 1$ condition of Appendix F.2, under which TPR at low FPR already remains at the random-guess level. By producing a consistent number ($|G_k|$) of synthetic samples and eliminating inter-client communication, this protocol prevents the leakage of ownership information between clients. Simultaneously, it ensures that raw embeddings and predictions never leave local storage during training, presenting only obfuscated representation-logit pairs to the server to safeguard data privacy while preserving representation utility.

Upon receiving local synthetic samples, the server identifies the contributing clients $\mathcal{CC}_i = \{m \mid \mathcal{N}_{m,k} \neq \emptyset\}$ and synthesizes a collective synthetic target by applying $\sigma(\cdot)$ to the averaged logits, $\bar{\mathbf{y}}_k^{(i)} = \sigma\big(\frac{1}{|\mathcal{CC}_i|}\sum_{m\in\mathcal{CC}_i} \tilde{\mathbf{l}}_{m,k}^{(i)}\big)$. This results in a graph-sample, where the synthetic client embeddings $\{\tilde{\mathbf{e}}_{m,k}^{(i)}\}_{m\in\mathcal{CC}_i}$ constitute the nodes and $\bar{\mathbf{y}}_k^{(i)}$ serves as the graph-level label. The detailed operational procedure is provided in Appendix A.4.

3.5 Relational Server Aggregation: ECG-GNN

We propose ECG-GNN, an architecture that fuses synthetic local representations by conditioning both message generation (ϕ) and gating (ψ) on the relational triplet $(\mathbf{h}_m^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{a}_{mj})$, where $\mathbf{h}_m^{(l)}$ and $\mathbf{h}_j^{(l)}$ are the target and neighbor node embeddings and $\mathbf{a}_{mj}$ encodes the raw feature overlap between clients m and j. Previous edge-aware models confine edge features to a single aspect, e.g., filters (Simonovsky & Komodakis, 2017) or attention scores (Gong & Cheng, 2019); conditioning both functions lets ECG-GNN capture inter-client interactions that such models under-represent. Here the set-level objective of (2) is realized as a per-sample loss ℓ over synthetic relational graphs, and to accommodate fluctuating client availability ECG-GNN operates on an active subset $\mathcal{AC}_i \subseteq \mathcal{CC}_i$ per sample i, with the synthetic representations and collective target updated over $\mathcal{AC}_i$ following Section 3.4.

3.5.1 Model Architecture

For each synthetic sample i, we construct the relational graph $\mathcal{R}^{(i)} = \big(\mathcal{AC}_i, \{\tilde{\mathbf{e}}_m^{(i)}\}_{m\in\mathcal{AC}_i}, \mathcal{E}^{(i)}, \mathbf{A}^{(i)}\big)$, where $\mathcal{E}^{(i)} = \mathcal{AC}_i \times \mathcal{AC}_i \setminus \{(m,m)\}$ explicitly defines a directed fully connected topology among the active clients for synthetic sample i. The edge attributes $\mathbf{A}^{(i)} = \{\mathbf{a}_{mj} \mid (m,j) \in \mathcal{E}^{(i)}\}$ represent the relational indicators mapping the feature space overlaps between client m and client j. Paired with the collective synthetic target $\bar{\mathbf{y}}^{(i)}$, this structure $\mathcal{R}^{(i)}$ serves as the topological template for training the ECG-GNN. Note that the relational graph for each sample is constructed over its active contributing clients $\mathcal{AC}_i \subseteq \mathcal{CC}_i$ rather than over all M clients, so the per-sample cost of message passing is $O(L_g \cdot |\mathcal{AC}_i|^2)$, treating the hidden and edge-feature dimensions as constants, where L_g denotes the number of message-passing layers of the ECG-GNN.

Node Encoding. To initialize relational reasoning, each node $m \in \mathcal{AC}_i$ is mapped into a unified latent space via a type-specific encoder $\mathbf{h}_m^{(0)} = \xi_{\tau(m)}(\tilde{\mathbf{e}}_m^{(i)})$, where $\xi_{\tau(m)}$ denotes the architecture tailored to the specific client type $\tau(m)$. In this work, we evaluate two distinct encoding strategies: (i) a homogeneous setup where all clients share a universal encoder ($\xi_{\tau(m)} = \xi_{global}$), and (ii) a heterogeneous setup where each client utilizes a specialized architecture to capture domain-specific characteristics.

Edge-Gated Message Passing. In the fully connected client graph, node update at layer $l \in \{0, \ldots, L_g - 1\}$ is driven by edge-conditioned functions $\phi^{(l)}$ (message generation) and $\psi^{(l)}$ (gating) across all other participating nodes $j \in \mathcal{AC}_i \setminus \{m\}$:

$$\mathbf{h}_m^{(l+1)} = \mathrm{LN}\left(\mathbf{h}_m^{(l)} + \sum_{j\in\mathcal{AC}_i\setminus\{m\}} g_{mj}^{(l)} \phi^{(l)}(\mathbf{z}_{mj})\right), \tag{5}$$

where $\mathrm{LN}(\cdot)$ denotes layer normalization and $g_{mj}^{(l)} = \frac{\exp(\psi^{(l)}(\mathbf{z}_{mj}))}{\sum_{k\in\mathcal{AC}_i\setminus\{m\}}\exp(\psi^{(l)}(\mathbf{z}_{mk}))}$ is the gating function conditioned on the relational triplet $\mathbf{z}_{mj} = [\mathbf{h}_m^{(l)} \,\|\, \mathbf{h}_j^{(l)} \,\|\, \mathbf{a}_{mj}]$, the concatenation of the target and neighbor node embeddings with the edge attribute.

Readout and Final Prediction. After L_g layers of edge-gated message passing, the final node representations $\{\mathbf{h}_m^{(L_g)} \mid m \in \mathcal{AC}_i\}$ are aggregated into a global representation $\mathbf{h}_{global} = \frac{1}{|\mathcal{AC}_i|}\sum_{m\in\mathcal{AC}_i}\mathbf{h}_m^{(L_g)}$ via mean pooling. The server-side network g then outputs the final logit vector through a linear projection, $\hat{\mathbf{l}}^{(i)} = g(\mathcal{R}^{(i)};\theta_g) = \mathbf{W}_{out}\mathbf{h}_{global} + \mathbf{b}_{out}$, with learnable $\mathbf{W}_{out}$ and $\mathbf{b}_{out}$. The final global prediction is subsequently obtained using the task-specific activation function $\sigma(\cdot)$ as $\hat{\mathbf{y}}^{(i)} = \sigma(\hat{\mathbf{l}}^{(i)})$. At inference, each holder $m \in \mathcal{CC}_n$ of a query sample forwards its embedding of the sample to the server, as in split or vertical FL inference; our privacy mechanisms target the training-time releases.

3.5.2 Server-side Training

The ECG-GNN is trained on the synthetic dataset $\tilde{\mathcal{D}}$ of Section 3.4 following Algorithm 7 (Appendix A.6). Rather than treating the synthetic samples as static, each sample i is instantiated per mini-batch by constructing $\mathcal{R}^{(i)}$ only over a randomly drawn active subset $\mathcal{AC}_i \subseteq \mathcal{CC}_i$, emulating the partial participation of real deployments. This serves a dual purpose: it makes the model robust to the stochastic absence of clients, and it acts as an implicit node-level dropout that prevents over-reliance on specific "hub" clients.

4 Experiments

4.1 Experimental Setup

Datasets and Tasks. Following prior HBFL work (Zhang et al., 2020b; Gao et al., 2023; Llasag Rosero et al., 2024), we partition standard benchmarks synthetically so that the sample and feature overlaps across clients can be controlled precisely. We evaluate on three modalities: FashionMNIST (Xiao et al., 2017), flattened into 784-dimensional tabular vectors ($M = 50$, the largest scale our hardware runs in parallel); CIFAR-100 (Krizhevsky & Hinton, 2009) as an image task in which each client holds a random $r \times r$ crop ($r \in [16, 32]$) of each image, resized to 32×32 ($M \in \{6, 8, 10\}$); and Multi-News (Fabbri et al., 2019) as a text task, where we transform the original summarization task into multi-label classification over 54 keyword labels extracted from the summaries ($M \in \{3, 10\}$; preprocessing and task design in Appendix B.1.1).

Experimental Configurations. We evaluate 12 settings (FM1–FM5, CI1–CI5, MN1–MN2) that vary sample and feature proportions; Table 5 in Appendix B.2 lists the clients, sample and feature ratios, and sharing strategy of each. Local parameter sharing follows Section 3.2: none in FM1–FM2 and Multi-News (independent encoders), group-wise selective sharing in FM3–FM5, and global sharing in CIFAR-100. Edge features are pixel-level overlaps for CIFAR-100 and, for Multi-News, membership of tokens ranked 35k–45k, the range of maximal cross-client variance (Appendices B.1.2–B.1.3).

Evaluation Metrics. For the single-label classification tasks (FashionMNIST and CIFAR-100), we report Accuracy. For the multi-label classification task on Multi-News, we utilize Micro-F1, mAP, and Hamming Loss (Schapire & Singer, 1999), applying a 0.2 prediction threshold to account for label sparsity. Results are averaged over 5 independent runs for FashionMNIST and CIFAR-100 and 3 for Multi-News, reflecting the higher computational cost of the latter. To ensure robustness, we re-randomize sample and feature assignments across clients for each experimental run.

Baselines. We compare our approach against three state-of-the-art baselines: HyFEM (Zhang et al., 2020b), FedHD (Gao et al., 2023), and FedLabSync (Llasag Rosero et al., 2024). For a fair comparison under the same parameter-sharing configurations as Fed-DREAM, we additionally evaluate FedLabSync-WA, a variant that adds periodic weight averaging of the shareable modules.

4.2 Implementation Details

Local Training and Encoder Architectures. Local models are a multi-layer perceptron (MLP) for FashionMNIST, ResNet-18 (He et al., 2016) for CIFAR-100, and mT5-small (Xue et al., 2021) for Multi-News, whose encoder hidden states are masked-mean-pooled into fixed-size vectors. Architectures, pooling, and the FM3–FM5 fusion layer are in Appendix C.1; training hyperparameters in Appendix C.2.

Structural and Privacy Parameters. Encoder depth L and minimum cluster size C are calibrated on a single randomly selected client of FM1, CI1, and MN1 via the protocols of Appendices A.2 and A.5, giving $L = 2, 3, 6$ and $C = 4, 5, 3$ for FashionMNIST, CIFAR-100, and Multi-News (knee points in Fig. 4); the Dirichlet parameter is $\alpha = 1.0$. The selection rationale, protocol hyperparameters, and proxy models are detailed in Appendix C.3.

Server-side ECG-GNN. Node encoding is client-specific for FashionMNIST and Multi-News, whose local encoders are not globally synchronized, and shared for CIFAR-100; Appendix E.2 compares the two strategies. Full details are in Appendix C.4.

Benchmarks and Environment. Implementations and hyperparameter configurations of all baselines are described in Appendix C.5. FedLabSync-WA is evaluated only where parameters are shared: it is omitted in FM1–FM2, where the overlapping feature groups are too small for stable weight averaging, and on Multi-News, where no parameters are shared; since Fed-DREAM likewise uses no parameter sharing in these settings (Section 4.1), the comparison there is between methods without parameter sharing. On Multi-News, HyFEM and FedHD are additionally excluded because their architectural and scalability constraints make them impractical for large Transformers, leaving FedLabSync as the sole baseline. All experiments use PyTorch on an Intel i9-12900K CPU and an NVIDIA RTX 4090 GPU. System costs – stage-wise wall-clock times, clustering-stage communication volume, and analytical estimates of the THE cryptographic workload – are profiled in Appendix G; the dominant cost is the one-time, offline secure distance aggregation (Stage 1 of Algorithm 2).

4.3 Comparison results with benchmark methods

Experimental Results on FashionMNIST and CIFAR-100. Table 1 reports accuracy at full participation ($P = 1.0$); partial-participation results ($P = 0.3$) are in Table 9, Appendix D. Two reference conditions, excluded from the best-result comparison and *Improv.*, bracket the results: *Local-only*, in which each client trains and infers on its own data, and *Oracle feature union*, in which a single model is trained on the zero-filled union of raw features (both defined in Appendix D).

Fed-DREAM achieves the best performance in every setting, leading FedLabSync by 0.5pp on average on FashionMNIST (0.6pp over FedLabSync-WA on FM3–FM5) and FedLabSync-WA by 2.8pp on CIFAR-100. Under partial participation ($P = 0.3$), Fed-DREAM retains its lead in every CIFAR-100 setting and improves by 1.8pp on average over full participation, widening the margin over the second-best benchmark to 2.6–8.4pp; we attribute the CIFAR-100 gain to the implicit node-level dropout of Section 3.5.2, which the benchmarks cannot exploit. Relative to the references, Fed-DREAM lifts accuracy over Local-only by 7–13pp on FashionMNIST and 13–23pp on CIFAR-100, whereas FedHD falls below Local-only on most FashionMNIST settings, a negative-transfer effect discussed in Appendix D; Fed-DREAM exceeds Oracle feature union in all FashionMNIST settings and in CI3 despite never assembling raw features in one place, indicating that distributed encoding, mixup augmentation, and relational fusion can outperform naively unioning zero-filled features. Standard deviations are reported in Appendix D.

Table 1: Mean classification accuracy on FashionMNIST and CIFAR-100 ($P = 1.0$). *Improv.* indicates the percentage point (pp) gain of Fed-DREAM over the second-best performing baseline. The *p*-val columns report the p-values of paired one-sided t-tests between Fed-DREAM and each benchmark, pooling all settings per dataset.

Method ($P = 1.0$)	**FashionMNIST**						**CIFAR-100**					
	FM1	**FM2**	**FM3**	**FM4**	**FM5**	***p*-val**	**CI1**	**CI2**	**CI3**	**CI4**	**CI5**	***p*-val**
Local-only	0.6501	0.6615	0.7532	0.7794	0.7942	-	0.2533	0.3034	0.3411	0.2752	0.3212	-
Oracle feature union	0.7070	0.7534	0.8062	0.8269	0.8405	-	0.4649	0.5055	0.5592	0.4491	0.4762	-
FedHD	0.5826	0.6606	0.7125	0.7804	0.7885	<0.001	0.3210	0.3615	0.3953	0.3123	0.3002	<0.001
HyFEM	0.7308	0.7829	0.8292	0.8324	0.8539	<0.001	0.3068	0.4067	0.4779	0.3285	0.3816	<0.001
FedLabSync	0.7492	0.7938	0.8343	0.8466	0.8565	<0.001	0.2599	0.3593	0.3998	0.3053	0.3578	<0.001
FedLabSync-WA	-	-	0.8326	0.8499	0.8605	<0.001	0.4081	0.4793	0.5338	0.3897	0.4285	<0.001
Fed-DREAM	**0.7493**	**0.7946**	**0.8371**	**0.8572**	**0.8659**	-	**0.4389**	**0.4952**	**0.5737**	**0.4213**	**0.4513**	-
Improv. (pp)	+0.01	+0.08	+0.28	+0.73	+0.54	-	+3.08	+1.59	+3.99	+3.16	+2.28	-

Experimental Results on Multi-News. Due to the significant computational burden of fine-tuning Transformer-based models, Multi-News is evaluated under limited client scales ($M \in \{3, 10\}$) with

full participation ($P = 1.0$). As summarized in Table 2, Fed-DREAM consistently outperforms the baselines across all metrics. The Oracle feature union reference is omitted for Multi-News: uniting the features of a sample amounts to concatenating all of its articles into a single sequence, which far exceeds the maximum sequence length T=512 of the local models, so the condition cannot be instantiated under the same architecture and configuration. Notably, our framework maintains its superiority even in scenarios where client-side parameter averaging is not performed. This confirms that the ECG-GNN successfully captures intricate cross-client correlations and achieves effective knowledge fusion without explicit weight synchronization. Standard deviations are in Appendix D.

Table 2: Mean performance on Multi-News ($P = 1.0$). Higher is better (↑) for Micro-F1 and mAP; lower is better (↓) for Hamming Loss. *Improv.* denotes the pp gain over FedLabSync (positive values indicate improvement). The *p-value* row reports paired one-sided t-tests against FedLabSync per metric, pooling all settings.

Method	**MN1** ($M = 3$)			**MN2** ($M = 10$)		
	Micro-F1 ↑	**mAP** ↑	**Hamming** ↓	**Micro-F1** ↑	**mAP** ↑	**Hamming** ↓
Local-only	0.3818	0.2825	0.0430	0.3168	0.2358	0.0466
FedLabSync	0.4176	0.3251	0.0416	0.3787	0.2707	0.0385
Fed-DREAM	**0.4358**	**0.3478**	**0.0312**	**0.3850**	**0.3050**	**0.0300**
Improv. (pp)	+1.82	+2.27	+1.04	+0.63	+3.43	+0.85
p-value	Micro-F1: 0.018		mAP: 0.016		Hamming: 0.006	

4.4 Ablation Study

To validate the contribution of each design element in Fed-DREAM, we evaluate the following ablated versions: *w/o edge feature* (using zero vectors for relational context), *w/o gate* (assigning constant weights to messages), *Mean-pool MLP* (replacing the ECG-GNN with an MLP over the mean-pooled client embeddings, i.e., server-side fusion without any relational structure), *w/o node connection* (omitting cross-client logit aggregation), and *w/o mixup* (training only on centroids). Detailed descriptions for each ablated version are provided in Appendix E.1.

Table 3 reports the lowest- and highest-accuracy setting of FashionMNIST and CIFAR-100 (full results with standard deviations and significance tests, together with a detailed comparison of the three fusion designs, are in Appendix E.1); every design component contributes significantly. Relational server-side fusion and multi-sample mixup are the dominant factors: ECG-GNN improves over a structure-free Mean-pool MLP in all ten settings, by 2.2pp on average and by 8.5/7.3pp on the low-overlap FM1–FM2, indicating that relational modeling matters most when feature overlap is small; edge features and gating contribute smaller but statistically significant improvements. The impact of gating mechanisms and node encoding strategies is analyzed in Appendix E.2.

Table 3: Ablation study results for representative experimental configurations.

Setting	**FM1**	**FM5**	**CI3**	**CI4**
Fed-DREAM	**0.7493**	**0.8659**	**0.5737**	**0.4213**
w/o edge feature	0.7459	0.8652	0.5681	0.4171
w/o gate	0.7487	0.8655	0.5661	0.4195
Mean-pool MLP	0.6641	0.8649	0.5532	0.4078
w/o node connection	0.6940	0.8569	0.4969	0.4028
w/o mixup	0.7447	0.8656	0.5248	0.3610

4.5 Privacy-Utility Evaluation

Utility cost of the privacy mechanisms. One would expect federated divisive clustering and multi-sample mixup to cost utility, since the server no longer sees the original embeddings. To measure this cost we train a *Raw-embedding* reference in which the ECG-GNN is trained directly on the original embeddings and labels (Tables 11 and 12, Appendix D). The cost turns out to be small: the reference exceeds Fed-DREAM by at most 2.9pp at P=1.0 (4.7pp at P=0.3), and in six of the twelve settings

at P=1.0 (all five FashionMNIST settings and MN1) Fed-DREAM is even better than the reference that sees unobfuscated data. We attribute the latter to the regularizing effect of multi-sample mixup: the server is trained on interpolated embeddings with softened targets rather than on the original points, which smooths its decision boundary in the same way as mixup does in centralized training.

Membership inference against the deployed system. An adversary with query access to the deployed system may attempt to infer whether a sample participated in training. We mount the loss-based attack of Yeom et al. (2018) at the system level – a sample $n \in \bigcup_m \mathcal{N}_m$ is a member if any client holds it – scoring each query by the deployed system's per-sample loss under an identical protocol for all systems. The audit covers FM1, FM4, CI1, and CI5 (both FashionMNIST sharing strategies and both CIFAR-100 client-scale extremes) and, in each setting, mounts the same attack against Fed-DREAM and the highest-utility benchmark of Table 1, FedLabSync-WA, or FedLabSync on FM1 where WA is structurally infeasible. Following Carlini et al. (2022), we report AUC and TPR at 1% and 0.1% FPR, the latter in Appendix F.1 together with the full protocol.

Membership inference results. In terms of AUC, Fed-DREAM leaks less than the benchmark in all four settings (Table 4), decisively on CIFAR-100 and consistently on FashionMNIST: the benchmark deploys local models fitted directly to the raw training samples, whose losses on those samples are markedly lower than on unseen ones, whereas the Fed-DREAM server never trains on original samples and retains far less per-sample memorization. At 1% FPR both systems stay near random guessing on FashionMNIST, while on CIFAR-100 Fed-DREAM attains the clearly lower TPR; at 0.1% FPR only ten non-member false positives are admissible per setting, which cannot separate the two systems (Appendix F.1). Combined with Tables 1 and 2, Fed-DREAM thus attains higher predictive utility than the benchmarks and lower membership leakage than the strongest of them.

Table 4: Membership inference vulnerability (mean $\pm$ std) of Fed-DREAM versus the highest-utility benchmark in each setting (FedLabSync on FM1, FedLabSync-WA otherwise). Random-guess levels are AUC 0.5 and TPR@1%FPR 0.01. TPR@0.1%FPR is reported in Appendix F.1.

Setting	AUC		TPR@1%FPR	
	Fed-DREAM	**Benchmark**	**Fed-DREAM**	**Benchmark**
FM1	$\mathbf{0.6145 \pm 0.0018}$	0.6449 ± 0.0020	0.0112 ± 0.0009	$\mathbf{0.0107 \pm 0.0016}$
FM4	$\mathbf{0.5768 \pm 0.0016}$	0.6302 ± 0.0029	$\mathbf{0.0099 \pm 0.0009}$	0.0114 ± 0.0003
CI1	$\mathbf{0.6094 \pm 0.0039}$	0.8490 ± 0.0060	$\mathbf{0.0151 \pm 0.0016}$	0.0191 ± 0.0015
CI5	$\mathbf{0.6060 \pm 0.0072}$	0.8776 ± 0.0148	$\mathbf{0.0157 \pm 0.0025}$	0.0271 ± 0.0025

Privacy–utility sensitivity to the minimum cluster size. The minimum cluster size C governs this privacy–utility trade-off, and our sensitivity analysis confirms the expected behavior: membership leakage (AUC) decreases monotonically with C in every setting, with TPR at low FPR already at the random-guess level from C=1, while the utility cost is dataset-dependent – accuracy declines with C on data-constrained CIFAR-100 but remains stable up to C=20 on FashionMNIST, where larger C can be adopted at little utility cost. Full sensitivity results are provided in Appendix F.2.

5 Conclusion

We proposed Fed-DREAM, an HBFL framework that bridges fragmented client views through federated divisive clustering, multi-sample representation-level mixup, and ECG-GNN. It consistently outperforms state-of-the-art benchmarks across tabular, image, and text modalities, every design element contributes significantly, and it attains lower membership leakage than the strongest benchmark rather than trading utility for privacy.

Limitation and Future Work. The per-sample cost of ECG-GNN grows quadratically in the number of contributing clients, a burden in large cross-silo federations; we will explore graph sparsification, e.g., restricting message passing to the neighbors with the largest feature overlap. The one-time Stage-1 distance aggregation also scales quadratically in per-client sample counts, although it runs offline before training and aligned ciphertext packing already reduces its cost by an order of magnitude. Finally, our privacy protection is established architecturally and validated empirically rather than through formal guarantees such as differential privacy; deriving such guarantees for the composed pipeline is an open direction we consider important.

AI USE STATEMENT

In this work, we did not use generative AI tools for developing the method or its conceptual framework, formulating claims, or implementing the proposed method or the baselines. We used `gpt-4o-mini` to transform the Multi-News summarization corpus into a multi-label classification dataset (keyword extraction and label assignment), as fully described in Appendix B. We further used generative AI tools to draft and edit parts of the manuscript for readability, to format LaTeX tables and references, to write scripts that aggregate experiment outputs into tables and plots, and to help phrase the interpretation of the reported results. We have reviewed all AI-assisted work: every AI-assisted passage was revised by the authors, and all reported numbers were cross-checked by the authors against the raw experiment outputs. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All datasets are publicly available, and their partitioning into the twelve HBFL settings is specified in Appendix B. Complete pseudo-code for every stage of Fed-DREAM is given in Appendix A; architectures, hyperparameters, and baseline implementations in Appendix C; the membership inference protocol and the system-cost profiling setup in Appendices F and G; and full results with standard deviations in Appendix D.

REFERENCES

Durmus Alp Emre Acar, Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Abhradeep Thakurta, and Florian Tramèr. Is private learning possible with instance encoding? In *IEEE Symposium on Security and Privacy (S&P)*, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 1897–1914. IEEE, 2022.

Timothy Castiglia, Anirban Das, Shiqiang Wang, and Stacy Patterson. Compressed-VFL: Communication-efficient learning with vertically partitioned data. In *International Conference on Machine Learning*, pp. 2738–2766, 2022.

Weijing Chen, Guoqiang Ma, Tao Fan, Yan Kang, Qian Xu, and Qiang Yang. SecureBoost+ : A high performance gradient boosting tree framework for large scale vertical federated learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2021.

Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. SecureBoost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6): 87–98, 2021.

Ivan B Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *International Workshop on Public Key Cryptography*, pp. 119–136, 2001.

Josep Domingo-Ferrer and Josep M Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14:189–201, 2002.

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Annual Meeting of the Association for Computational Linguistics*, pp. 1074–1084. Association for Computational Linguistics, 2019.

Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography, 4th International Conference*, pp. 90–104, 2000.

Dashan Gao, Yang Liu, Anbu Huang, Ce Ju, Han Yu, and Qiang Yang. Privacy-preserving heterogeneous federated transfer learning. In *IEEE International Conference on Big Data*. IEEE, 2019.

Haiqing Gao, Songyang Ge, and Tsung Hui Chang. FedHD: Communication-efficient federated learning from hybrid data. *Journal of the Franklin Institute*, 360(12):8416–8454, 2023.

Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

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

Yuanqin He, Yan Kang, Xinyuan Zhao, Jiahuan Luo, Lixin Fan, Yuxing Han, and Qiang Yang. A hybrid self-supervised learning framework for vertical federated learning. *IEEE Transactions on Big Data*, 11(5):2210–2223, 2025.

Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. InstaHide: Instance-hiding schemes for private distributed learning. In *International Conference on Machine Learning*, 2020.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2): 1–210, 2021.

Yan Kang, Yang Liu, and Xinle Liang. FedCVT: Semi-supervised vertical federated learning with cross-view training. *ACM Transactions on Intelligent Systems and Technology*, 13(4), 2022.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143, 2020.

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York, 1990.

Afsana Khan, Marijn ten Thij, and Anna Wilbik. Communication-efficient vertical federated learning. *Algorithms*, 15(8):273, 2022.

Afsana Khan, Marijn ten Thij, Guangzhi Tang, and Anna Wilbik. HybridFL: A federated learning approach for financial crime detection. *arXiv preprint arXiv:2602.19207*, 2026.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Machine Learning and Systems*, 2020.

Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35(4):70–82, 2020.

Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. FATE: An industrial grade platform for collaborative learning with data protection. *Journal of Machine Learning Research*, 22:1–6, 2021.

Raúl Llasag Rosero, Catarina Silva, Bernardete Ribeiro, and Bruno F. Santos. Label synchronization for hybrid federated learning in manufacturing and predictive maintenance. *Journal of Intelligent Manufacturing*, 35(8):4015–4034, 2024.

Raúl Llasag Rosero, Catarina Silva, Bernardete Ribeiro, Melania Albisser, Martin Brutsche, and Manuel Arias Chao. Label synchronization strategies for hybrid federated learning. *Reliability Engineering and System Safety*, 256, 2025.

H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Dianwen Ng, Xiang Lan, Melissa Min-Szu Yao, Wing P. Chan, and Mengling Feng. Federated learning: a collaborative effort to achieve better medical imaging models for individual sites that have small labelled datasets. *Quantitative Imaging in Medicine and Surgery*, 11(2):852–857, 2021.

Kai Wang Ng, Guo-Liang Tian, and Man-Lai Tang. *Dirichlet and Related Distributions*. Wiley, 2011.

Tom Overman, Garrett Blum, and Diego Klabjan. A primal-dual algorithm for hybrid federated learning. In *AAAI Conference on Artificial Intelligence*, 2024.

Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1592, pp. 223–238. Springer-Verlag, 1999.

Ville Satopää, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *International Conference on Distributed Computing Systems*, pp. 166–171, 2011.

Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

S. Sharma, C. Xing, Y. Liu, and Y. Kang. Secure and efficient federated transfer learning. In *IEEE International Conference on Big Data*, pp. 2569–2576. IEEE, 2019.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, 2021.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, pp. 268–282. IEEE, 2018.

Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning. In *USENIX Annual Technical Conference (ATC)*, pp. 493–506, 2020a.

Xinwei Zhang, Wotao Yin, Mingyi Hong, and Tianyi Chen. Hybrid federated learning: Algorithms and implementation. *arXiv preprint arXiv:2012.12420*, 2020b.

A Detailed Algorithms

A.1 Local Optimization and Encoder Learning

In this section, we provide the detailed pseudo-code for the generalized local optimization phase discussed in Section 3.2. Algorithm 1 outlines the precise procedure for our localized training with conditional communication.

To accommodate highly heterogeneous client architectures, varying feature overlaps, and strict privacy constraints, the algorithm partitions the local model parameters into a globally shared component (Θ) and a strictly private component (θ_m^{pr}). When modules are structurally shareable and clients consent to collaborate, the server broadcasts the globally shared parameters Θ at the beginning of each round. Each selected client then extracts its corresponding sub-modules, assembles the full local model, and performs standard local optimization. Subsequently, the updated shared modules are uploaded to the server for a data-size-weighted aggregation, following the principles of federated averaging (McMahan et al., 2017).

If parameter sharing is precluded by structural constraints or selective non-participation to mitigate negative transfer, the algorithm bypasses synchronization and defaults to standalone local training.

Algorithm 1 Detailed Conditional Federated Averaging for HBFL

Require: M: total number of clients, E: local epochs, η: learning rate, T: communication rounds.
Require: $\{\mathcal{K}_m\}_{m=1}^{M}$: set of shared module indices possessed by each client m.
Server Configuration: Initialize global parameter set $\Theta_0 = \{\Theta^{(k)}\}_{k \in \bigcup \mathcal{K}_m}$.
Client Configuration: Initialize local private parameters $\{\theta_m^{pr}\}_{m=1}^{M}$.
for round $t = 1$ to T **do**
  Identify active clients $\mathcal{S}_t \subseteq \{1, \dots, M\}$ for the current round.
  Server broadcasts relevant modules $\{\Theta^{(k)}\}_{k \in \bigcup_{m \in \mathcal{S}_t} \mathcal{K}_m}$ to respective clients.
  for each client $m \in \mathcal{S}_t$ **in parallel do**
    Local Initialization:
    Fetch possessed shared modules: $\theta_m^{sh} \leftarrow \{\Theta^{(k)}\}_{k \in \mathcal{K}_m}$.
    Assemble full local model parameters: $\theta_m \leftarrow \theta_m^{sh} \cup \theta_m^{pr}$.
    Local Training:
    for epoch $e = 1$ to E **do**
      for mini-batch $\mathcal{B} \subset \mathcal{D}_m$ **do**
        $\mathbf{e}_{m,n} \leftarrow f_m^{emb}(\mathbf{x}_{m,n}; \theta_m^{emb})$, $\hat{\mathbf{y}}_{m,n} \leftarrow \sigma\big(f_m^{head}(\mathbf{e}_{m,n}; \theta_m^{head})\big)$.
        $\mathcal{L}_m \leftarrow \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \ell(\hat{\mathbf{y}}_{m,n}, \mathbf{y}_n)$; $\theta_m \leftarrow \theta_m - \eta \nabla_{\theta_m} \mathcal{L}_m$.
      end for
    end for
    Upload: Updated shared modules $\{\theta_m^{sh(k)}\}_{k \in \mathcal{K}_m}$ and $|\mathcal{D}_m|$ to server.
  end for
  Server Aggregation:
  for each module index $k \in \bigcup_{m \in \mathcal{S}_t} \mathcal{K}_m$ **do**
    Identify clients owning module k: $COM_k = \{m \in \mathcal{S}_t \mid k \in \mathcal{K}_m\}$.
    $\Theta^{(k)} \leftarrow \sum_{m \in COM_k} \frac{|\mathcal{D}_m|}{\sum_{j \in COM_k} |\mathcal{D}_j|} \theta_m^{sh(k)}$. {Module-specific weighted averaging}
  end for
end for
return Optimized parameters $\{\theta_m = \{\Theta^{(k)}\}_{k \in \mathcal{K}_m} \cup \theta_m^{pr}\}_{m=1}^{M}$ for individual clients

A.2 Encoder Depth Selection Protocol

In our framework, determining the optimal boundary (i.e., depth L) that separates the local feature encoder from the local prediction head within a client's complete local network is a critical decision. Intuitively, one might assume that a deeper encoder inherently provides higher privacy by applying more complex, non-linear transformations to the raw data before sharing the representations. However, true privacy is fundamentally linked to the size of the ambiguity set—the set of distinct raw inputs that could map to the same shared embedding. If only a small number of raw inputs can produce a specific embedding, an adversary can more easily pinpoint the original raw data. Conversely, a larger

pool of ambiguity set enhances privacy by increasing the ambiguity of the shared representation, making it harder for an attacker to isolate the true input from other candidates.

To quantify this, we propose a privacy score based on the concept of diameter, defined as the maximum distance from a target sample in the input space that yields an identical embedding. Let $\mathbf{x}_{m,n}$ represent the target sample, and $\mathbf{e}_{m,n}^{j}$ denote the original embedding produced by the local encoder truncated at layer j, denoted as $f_m^{emb,j}$. To evaluate the robustness of this diameter against potential vulnerabilities, we assume a practical threat model where the encoder parameters are subject to perturbation, resulting in a perturbed encoder $\tilde{f}_m^{emb,j}$ parameterized by $\tilde{\theta}_m^{emb,j}$.

To find the worst-case privacy score s_m^j across the client's local dataset $\mathcal{N}_m$, we formulate the problem as a bi-level optimization using Lagrangian relaxation with a large penalty weight $\delta \gg 0$:

$$s_m^j = \min_{\tilde{\theta}_m^{emb,j}} \sum_{n \in \mathcal{N}_m} \max_{\bar{\mathbf{x}}_{m,n}} \left(\|\bar{\mathbf{x}}_{m,n} - \mathbf{x}_{m,n}\|^2 - \delta \|\tilde{f}_m^{emb,j}(\bar{\mathbf{x}}_{m,n}; \tilde{\theta}_m^{emb,j}) - \mathbf{e}_{m,n}^{j}\|^2 \right). \tag{6}$$

Specifically, to calculate s_m^j, we employ an alternating optimization strategy that iteratively updates the perturbed input $\bar{\mathbf{x}}_{m,n}$ and the perturbed parameters $\tilde{\theta}_m^{emb,j}$. Subsequently, we apply stochastic gradient ascent/descent to alternate between the following two objectives:

1. **Inner Maximization (Diameter Calculation)**: We aim to identify a sample $\bar{\mathbf{x}}_{m,n}^*$ that defines the diameter of the ambiguity set by maximizing the distance from the original sample while rigorously ensuring the embedding constraint is met. This process effectively calculates the maximum radius in the input space within which any sample yields an identical embedding. To solve for this diameter, we formulate the following maximization problem:

$$\bar{\mathbf{x}}_{m,n}^* \leftarrow \arg\max_{\bar{\mathbf{x}}_{m,n}} \left(\|\bar{\mathbf{x}}_{m,n} - \mathbf{x}_{m,n}\|^2 - \delta \|\tilde{f}_m^{emb,j}(\bar{\mathbf{x}}_{m,n}; \tilde{\theta}_m^{emb,j}) - \mathbf{e}_{m,n}^{j}\|^2 \right), \quad \forall n \in \mathcal{N}_m. \tag{7}$$

If the optimization is initialized directly at the original sample $\mathbf{x}_{m,n}$, the initial distance penalty is zero and the gradients vanish, preventing the updates. To prevent this and successfully initiate the procedure, we initialize the candidate input $\bar{\mathbf{x}}_{m,n}$ by adding a small perturbation drawn from a Laplace distribution to $\mathbf{x}_{m,n}$. Starting from this Laplace-perturbed state, we apply gradient ascent to iteratively explore the input space and determine the maximum distance that preserves the embedding, thereby calculating the diameter of the ambiguity set.

2. **Outer Minimization (Worst-Case Parameter Perturbation)**: Using the diameter-defining samples $\bar{\mathbf{x}}_{m,n}^*$ identified in the inner step, we optimize the encoder's parameters to minimize the resulting privacy score. This step searches for a specific parameter perturbation $\tilde{\theta}_m^{emb,j}$ that most severely shrinks the diameter of the ambiguity set, thereby representing a conservative, worst-case estimate of the diameter. By finding the parameters that make the embedding mapping most sensitive to the input, we ensure that the final score s_m^j reflects a conservative empirical estimate of the privacy level under potential parameter variations:

$$\tilde{\theta}_m^{emb,j} \leftarrow \arg\min_{\tilde{\theta}_m^{emb,j}} \sum_{n \in \mathcal{N}_m} -\delta \|\tilde{f}_m^{emb,j}(\bar{\mathbf{x}}_{m,n}^*; \tilde{\theta}_m^{emb,j}) - \mathbf{e}_{m,n}^{j}\|^2. \tag{8}$$

While a higher score s_m^j indicates enhanced privacy (a larger ambiguity set), we generally observe a decline in this score as the number of hidden layers increases. With each additional layer, the mapping becomes highly sensitive; thus, parameter perturbations drastically shrink the feasible diameter. However, from an architectural standpoint, deeper layers also subject the raw data to successive non-linear transformations, highly entangling the representations and inherently increasing the difficulty of direct inversion. Because of this structural complexity, a gradual decline in s_m^j as j increases is considered an acceptable trade-off rather than a critical vulnerability.

Consequently, determining the optimal encoder depth L does not mean simply selecting the shallowest layer with the highest score. Instead, it requires identifying the critical threshold where this privacy degradation abruptly accelerates. By analyzing the decreasing curve of s_m^j across varying depths, we seek the "knee" point—the exact layer just before a precipitous drop in the privacy score occurs. To systematically and quantitatively detect this point of maximum curvature, we employ the Kneedle algorithm (Satopää et al., 2011). By setting L at this detected knee point, our protocol ensures that the local encoder is deep enough to extract highly entangled, meaningful representations, while preemptively avoiding the severe privacy penalty induced by extreme over-sensitivity.

A.3 Detailed Procedure for Federated Divisive Clustering

In Section 3.3, we outline the high-level workflow of our privacy-preserving federated divisive clustering. Due to space constraints in the main text, the comprehensive operational details of the four primary stages are elaborated below. These descriptions provide the necessary context for the memory-efficient communication and the cryptographic privacy guarantees, followed by the complete pseudo-codes (Algorithms 2, 3, and 4).

Central to our framework is a custom size-constrained divisive clustering subroutine (Algorithm 3), which is repeatedly invoked across the local and global stages. This subroutine adapts the classic farthest-first traversal heuristic (Gonzalez, 1985; Kaufman & Rousseeuw, 1990) to efficiently bisect the sample sets based on a pivot-based split. To strictly enforce the privacy-preserving bounds of $[C, 2C-1]$, we incorporate a distance-based greedy borrowing mechanism. This strategy ensures that any bisected sub-group falling below the minimum threshold C is augmented with the nearest neighbors from its larger counterpart, a size-constrained borrowing step adapted from microaggregation (Domingo-Ferrer & Mateo-Sanz, 2002). The federated orchestration utilizing this subroutine proceeds as follows:

1) Secure Distance Aggregation: To avert the risk of memory exhaustion inherently associated with simultaneously computing and storing a massive global distance matrix, the server orchestrates the computation by requesting specific batches of sample index pairs. Participating clients compute pairwise distances between their local embeddings $f_m^{emb}(\mathbf{x}_{m,n})$, encrypt them using a collaboratively generated public key, and upload them to the server. Employing a secure aggregation protocol based on the THE scheme, the server homomorphically computes the aggregated sums without decrypting the individual inputs. The server then broadcasts these aggregated ciphertexts back to the clients, who collaboratively compute partial decryptions using their secret key shares. The server combines these shares to derive the exact aggregated sums. Utilizing its explicit mapping of local indices, the server divides these sums by the number of observing clients to calculate the precise averaged global distance matrix D_G in sequential batches, ensuring that individual local distance matrices remain confidential for pairs observed by at least two clients ($|\mathcal{V}_{i,j}| \geq 2$); for singleton pairs the aggregate equals the sole observer's entry, and enforcing a minimum of two contributors per pair would close this leakage at the cost of a sparser global distance matrix. Finally, the server distributes the corresponding aggregated distance segments back to the clients.

2) Local Clustering & Merging: Each client assembles these received pieces to construct the complete distance matrix D_G specific to its local indices. Utilizing this assembled matrix, clients independently execute the size-constrained divisive clustering (Algorithm 3) to construct local clusters. Subsequently, they upload only the derived cluster medoid indices and their associated membership mappings to the server. Upon receiving the local results, the server first consolidates the clusters by merging membership lists that share identical medoids across clients.

3) Global Medoid Clustering: The server performs a higher-level divisive clustering on the pooled medoids. To circumvent the memory overhead of maintaining the full D_G, the server does not store the global matrix in its entirety; instead, once the medoids are identified, it dynamically requests the pairwise distances between the underlying members of target medoid pairs. Note that in Stage 1 the server redistributes the decrypted distance segments to their observing clients and does not retain the full global matrix; hence, entries required by the medoid-merging stage are retrieved in plaintext from the clients that received them in Stage 1, which reveals nothing beyond what those clients already hold. The distance between any two medoids is then computed as the average cross-cluster distance, calculated exclusively over the valid inter-cluster sample pairs that possess observable distance entries, as detailed in Algorithm 4. If no such co-occurring samples exist between the two medoid memberships, a large penalty is assigned to discourage the grouping of structurally disconnected clusters. Note that Stage 3 applies C to the medoids, so a medoid group contains far more than $2C-1$ samples; the sample-level bound is restored in Stage 4, which re-partitions each group.

4) Refinement: Finally, the server aggregates the underlying samples of these grouped medoids and applies a recursive refinement step, invoking the divisive clustering subroutine on any oversized clusters. Consistent with the memory-efficient design, the pairwise distances required for this recursive partitioning are retrieved on demand from their holders in the same way. This step strictly enforces the $[C, 2C-1]$ size boundary across all final global clusters.

Algorithm 2 Privacy-Preserving Federated Divisive Clustering

Require: Target minimum cluster size C, Global index set $\mathcal{N}$, Local index sets $\{\mathcal{N}_m\}_{m=1}^M$, Local embeddings $\{f_m^{emb}(\mathbf{x}_{m,n})\}_{n\in\mathcal{N}_m, m\in\{1,\dots,M\}}$

{Subroutines DIVISIVECLUSTERING and AVGDIST refer to Algorithms 3 and 4.}
[Key Setup] Clients collaboratively generate a public key (pk) and secret key shares $\{sk_m\}_{m=1}^M$ using a THE scheme. Server acts as an aggregator for ciphertexts encrypted under pk.
[Stage 1: Secure Global Distance Aggregation]
{Iterate over all index pairs (i,j) appearing in at least one client's local set}
for each batch of pairs $(i,j) \in \bigcup_{m=1}^M (\mathcal{N}_m \times \mathcal{N}_m)$ **do**
  Server initializes encrypted sum $\widetilde{D}_{sum}[i,j] \leftarrow \text{Enc}_{pk}(0)$.
  Server: Identify observer set $\mathcal{V}_{i,j} = \{m \mid i,j \in \mathcal{N}_m\}$.
  for each client $m \in \mathcal{V}_{i,j}$ **in parallel do**
    Client m computes local distance $D_m[i,j]$ between its local embeddings $f_m^{emb}(\mathbf{x}_{m,i})$ and $f_m^{emb}(\mathbf{x}_{m,j})$.
    Send $\widetilde{D}_m[i,j] \leftarrow \text{Encrypt}(D_m[i,j], pk)$ to Server.
  end for
  Server: Homomorphically aggregate and compute plaintext via collaborative decryption:
    $\widetilde{D}_{sum}[i,j] \leftarrow \sum_{m\in\mathcal{V}_{i,j}} \widetilde{D}_m[i,j]$.
    $D_{sum}[i,j] \leftarrow \text{CollaborativeDecrypt}(\widetilde{D}_{sum}[i,j], \{sk_m\})$.
    $D_G[i,j] \leftarrow D_{sum}[i,j]/|\mathcal{V}_{i,j}|$. {Global averaged distance}
  Server: Send $D_G[i,j]$ to each client $m \in \mathcal{V}_{i,j}$ for local matrix construction.
end for
[Stage 2: Local Clustering & Medoid Selection]
for each client $m \in \{1,\dots,M\}$ **in parallel do**
  Construct local distance matrix $D_G[\mathcal{N}_m, \mathcal{N}_m]$ from aggregated global entries.
  $\mathcal{M}_m, \mathcal{MS}_m \leftarrow \text{DIVISIVECLUSTERING}(\mathcal{N}_m, D_G[\mathcal{N}_m,\mathcal{N}_m], C)$.
  $\{G_{m,k}\}_{k\in\mathcal{M}_m} \leftarrow \{\mathcal{MS}_m[\mu] : \mu \in \mathcal{M}_m\}$. {Convert the returned mapping into membership sets}
  Send local medoids $\mathcal{M}_m$ and membership sets $\{G_{m,k}\}_{k\in\mathcal{M}_m}$ to Server.
end for
[Stage 3: Global Medoid Clustering]
Server: Construct global medoid set $\mathcal{M}_G = \bigcup_m \mathcal{M}_m$.
Server: For each medoid $k \in \mathcal{M}_G$, map its total assigned samples $\mathcal{S}_k = \bigcup_{m:k\in\mathcal{M}_m} G_{m,k}$.
Server: $\mathcal{M}_H, \mathcal{MS}_H \leftarrow$DIVISIVECLUSTERING$(\mathcal{M}_G, \text{inter-medoid distances}, C)$.
Server: $\{\mathcal{H}_v\}_v \leftarrow \{\mathcal{MS}_H[\mu] : \mu \in \mathcal{M}_H\}$.
{Inter-medoid distances are computed on-demand using AVGDIST based on D_G}
[Stage 4: Sample Aggregation and Final Refinement]
Initialize final global cluster set $\mathcal{G} \leftarrow \emptyset$.
for each medoid cluster $\mathcal{H}_v$ **do**
  Aggregate samples associated with medoids in the cluster: $\mathcal{X}_v \leftarrow \bigcup_{k\in\mathcal{H}_v} \mathcal{S}_k$.
  if $|\mathcal{X}_v| \geq 2C$ **then**
    $\mathcal{M}_v, \mathcal{MS}_v \leftarrow \text{DIVISIVECLUSTERING}(\mathcal{X}_v, D_G[\mathcal{X}_v,\mathcal{X}_v], C)$.
    $\{G_{v,j}\}_{j\in\mathcal{M}_v} \leftarrow \{\mathcal{MS}_v[\mu] : \mu \in \mathcal{M}_v\}$.
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{G_{v,j}\}_{j\in\mathcal{M}_v}$.
  else
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{X}_v\}$.
  end if
end for
return Global clusters $\mathcal{G} = \{G_1, G_2, \dots, G_K\}$, where each $|G_k| \in [C, 2C-1]$

A.4 DETAILS OF MULTI-SAMPLE REPRESENTATION-LEVEL MIXUP

This section provides the operational details of the federated multi-sample representation-level mixup protocol, as summarized in Algorithm 5. This phase transforms private local embeddings into obfuscated synthetic representations while handling the inherent sample heterogeneity of HBFL.

Privacy and Ownership Protection: The interpolation is performed locally on the client side. This ensures that the raw local embeddings and their corresponding predictions never leave the client's storage. The use of a Dirichlet distribution ensures that each transmitted representation is a weighted composite of multiple samples, leaving the server unable to uniquely determine the individual embeddings underlying a synthetic representation, even if it possesses the complete client-sample

Algorithm 3 Size-Constrained Divisive Clustering (DIVISIVECLUSTERING)

Require: Index set $\mathcal{I}$ (where $|\mathcal{I}| \geq 2C$), Distance matrix D_G, Target minimum size C
Initialize a FIFO queue $\mathcal{Q}$ and Enqueue($\mathcal{I}$).
Initialize local cluster collection $\mathbf{P} \leftarrow \emptyset$. {Using $\mathbf{P}$ to avoid clash with global $\mathcal{G}$}
while $\mathcal{Q}$ is not empty **do**
  $\mathcal{W} \leftarrow$ Dequeue($\mathcal{Q}$). {$\mathcal{W}$ is the current working subset}
  if $C \leq |\mathcal{W}| < 2C$ **then**
    $\mathbf{P} \leftarrow \mathbf{P} \cup \{\mathcal{W}\}$.
    continue
  end if
  {Pivot-based split to find two sub-groups}
  Find farthest pair in $\mathcal{W}$: $z_1, z_2 \leftarrow \arg\max_{x,y \in \mathcal{W}} D_G[x, y]$.
  $\mathcal{U}_1 \leftarrow \{x \in \mathcal{W} \mid D_G[x, z_1] < D_G[x, z_2]\}$.
  $\mathcal{U}_2 \leftarrow \mathcal{W} \setminus \mathcal{U}_1$.
  {Size constraint enforcement via distance-based borrowing}
  if $|\mathcal{U}_1| < C$ **or** $|\mathcal{U}_2| < C$ **then**
    Identify smaller group $\mathcal{U}_{min}$ and larger group $\mathcal{U}_{max}$ from $\{\mathcal{U}_1, \mathcal{U}_2\}$.
    Let z_{min} be the pivot associated with $\mathcal{U}_{min}$.
    Sort elements $x \in \mathcal{U}_{max}$ by $D_G[x, z_{min}]$ in ascending order.
    $k \leftarrow C - |\mathcal{U}_{min}|$. {Number of points needed to meet constraint}
    Move the first k elements from $\mathcal{U}_{max}$ to $\mathcal{U}_{min}$.
    Update $\mathcal{U}_1, \mathcal{U}_2$ using the adjusted $\mathcal{U}_{min}, \mathcal{U}_{max}$.
  end if
  Enqueue($\mathcal{Q}, \mathcal{U}_1$), Enqueue($\mathcal{Q}, \mathcal{U}_2$).
end while
{Compute Medoids and Memberships}
Initialize medoid set $\mathcal{M} \leftarrow \emptyset$, membership mapping $\mathcal{MS} \leftarrow \{\}$.
for each cluster $c \in \mathbf{P}$ **do**
  Find medoid: $\mu_c \leftarrow \arg\min_{x \in c} \sum_{y \in c} D_G[x, y]$.
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu_c\}$.
  $\mathcal{MS}[\mu_c] \leftarrow c$. {Mapping medoid to its cluster members}
end for
return $\mathcal{M}, \mathcal{MS}$

Algorithm 4 Average Distance Between Medoids with Batch Retrieval (AVGDIST)

Require: Medoids m_1, m_2, Global membership dictionary $\mathcal{MS}$
Server: Identify the set of all pairs $\mathcal{P} = \{(x, y) \mid x \in \mathcal{MS}[m_1], y \in \mathcal{MS}[m_2]\}$.
Server: Request the global distance entries for $\mathcal{P}$ from clients in sequential batches.
Clients: Return the requested D_G entries in plaintext (decrypted in Stage 1).
Server:
  Initialize sum $\leftarrow 0$, count $\leftarrow 0$.
  {Calculate exclusively over valid inter-cluster sample pairs}
for each pair $(x, y) \in \mathcal{P}$ **do**
  if $D_G[x, y]$ was returned (i.e., observed by at least one client) **then**
    sum $\leftarrow$ sum $+ D_G[x, y]$.
    count $\leftarrow$ count $+ 1$.
  end if
end for
if count > 0 **then**
  return sum/count
else
  return Large penalty for structurally disconnected sets
end if

membership information (i.e., the sets $\{\mathcal{N}_m\}_{m=1}^{M}$). Since the mixup weights are sampled locally by each client and never disclosed, the server's observation constitutes a blind source separation problem with undisclosed non-negative weights; the membership leakage of the deployed system is audited empirically in Section 4.5.

Server-Blind Weight Generation: Crucially, the mixing weights themselves must be hidden from the server: if the server generated or observed the exact realized weights, then for a client m contributing

$c = |\mathcal{N}_{m,k}|$ local samples to cluster G_k, collecting c or more synthetic embeddings $\tilde{\mathbf{e}}_{m,k}^{(i)}$ with known, linearly independent weight vectors would let the server solve the corresponding linear system exactly for every underlying raw embedding $f_m^{\text{emb}}(x_{m,n})$ – entirely defeating the one-way property of Section 3, regardless of how the mixup itself is computed. We therefore have each client m sample its own $\{\boldsymbol{\lambda}^{(i)}\}_{i=1}^{|G_k|}$ over $\mathcal{N}_{m,k}$ from its own local random source, independently of every other client and of the server. This requires no key material, no inter-client coordination, and no protocol beyond a local call to a Dirichlet sampler, since the realized weights are never transmitted in any form.

Relation to mixup-based obfuscation. InstaHide (Huang et al., 2020) mixes raw images with random coefficients and was inverted by Carlini et al. (2021), who regressed the images from the many mixtures released for each of them. Fed-DREAM shares this many-mixtures structure, since each client releases $|G_k|$ mixtures of the same at most $2C - 1$ embeddings, so in the worst case a similar regression might approximate the embeddings even though the weights are never disclosed. Reaching the raw features from such an estimate is another matter: the mixing acts on the output of a non-injective encoder, so a recovered embedding corresponds to an entire ambiguity set of inputs (Appendix A.2) rather than to a single sample.

Algorithm 5 Federated Multi-sample Representation-level Mixup

Require: Global clusters $\{G_k\}_{k=1}^{K}$, Local embeddings $\{f_m^{emb}(\mathbf{x}_{m,n})\}_{n\in\mathcal{N}_m, m\in\{1,\dots,M\}}$, Local predictions $\{f_m^{head}(f_m^{emb}(\mathbf{x}_{m,n}))\}_{n\in\mathcal{N}_m, m\in\{1,\dots,M\}}$, Dirichlet parameter α

Initialize global synthetic dataset $\tilde{\mathcal{D}} \leftarrow \emptyset$.
for each cluster G_k **do**
  for each participating client m **in parallel do**
    Client m**:** Independently sample $\{\boldsymbol{\lambda}^{(i)}\}_{i=1}^{|G_k|}$ i.i.d. from Dir$(\alpha, \dots, \alpha)$ over its own local index set $\mathcal{N}_{m,k}$, where each $\boldsymbol{\lambda}^{(i)} \in \mathbb{R}^{|\mathcal{N}_{m,k}|}$.
    for $i = 1$ **to** $|G_k|$ **do**
      $\tilde{\mathbf{e}}_{m,k}^{(i)} \leftarrow \sum_{n\in\mathcal{N}_{m,k}} \lambda_n^{(i)} f_m^{emb}(\mathbf{x}_{m,n})$.
      $\tilde{\mathbf{l}}_{m,k}^{(i)} \leftarrow \sum_{n\in\mathcal{N}_{m,k}} \lambda_n^{(i)} f_m^{head}(f_m^{emb}(\mathbf{x}_{m,n}))$.
      Send $(\tilde{\mathbf{e}}_{m,k}^{(i)}, \tilde{\mathbf{l}}_{m,k}^{(i)})$ to the Server.
    end for
  end for
  Server: For each synthetic sample $i \in \{1, \dots, |G_k|\}$, aggregate received contributions.
    $\mathcal{CC}_i \leftarrow \{m \mid \mathcal{N}_{m,k} \neq \emptyset\}$. {Contributing clients set}
    Collective synthetic target: $\bar{\mathbf{y}}_k^{(i)} \leftarrow \sigma\big(\frac{1}{|\mathcal{CC}_i|} \sum_{m\in\mathcal{CC}_i} \tilde{\mathbf{l}}_{m,k}^{(i)}\big)$.
    $\tilde{\mathcal{D}} \leftarrow \tilde{\mathcal{D}} \cup \{(\mathcal{CC}_i, \{\tilde{\mathbf{e}}_{m,k}^{(i)}\}_{m\in\mathcal{CC}_i}, \bar{\mathbf{y}}_k^{(i)})\}$.
end for
return $\tilde{\mathcal{D}}$

A.5 Automated Selection of Minimum Cluster Size (C)

This section details the operational protocol for determining the global minimum cluster size C. Consistent with the encoder depth selection protocol in Appendix A.2, we adopt a systematic approach to balance privacy and utility by identifying the critical threshold via curvature analysis on the performance-distortion curve.

A.5.1 Protocol Overview and Data Partitioning

We adopt a representative-client-based calibration strategy, balancing structural consistency across the federation with computational efficiency. A single client is designated by the server as the representative to perform the calibration locally. Utilizing a pre-configured fraction $s \in (0, 1)$, the representative client partitions its private data $\mathcal{D}_{rep}$ into training ($\mathcal{D}_{train}$) and validation ($\mathcal{D}_{val}$) subsets, where $\mathcal{D}_{val}$ serves to quantify model generalization across distinct obfuscation degrees.

A.5.2 Metric Evaluation and Kneedle-based Selection

The calibration involves an iterative search over a candidate search space $\mathcal{CSS} = \{1, \dots, C_{max}\}$. For each $C \in \mathcal{CSS}$, the representative client adaptively partitions its local embeddings into K_C

groups $\{G_k\}_{k=1}^{K_C}$ using size-constrained divisive clustering. We then evaluate the trade-off between the following two metrics:

- **Average Intra-cluster Distance** ($\Gamma(C)$): This measures the clustering cohesion within the local latent space. It is computed as the mean distance between training embeddings $f^{emb}(\mathbf{x} \in \mathcal{D}_{train})$ and their respective cluster medoids within the partition $\{G_k\}_{k=1}^{K_C}$.
- **Utility Performance Metric** ($\mathbb{U}(C)$): This represents the predictive performance evaluated on $\mathcal{D}_{val}$. To simulate the proposed obfuscation, a synthetic dataset $\tilde{\mathcal{D}}$ is first generated by applying the representation-level mixup protocol locally within each cluster G_k. This process yields local synthetic embeddings and their corresponding targets, rather than the full multi-client graph structures used during actual server training. Subsequently, a proxy model is trained on $\tilde{\mathcal{D}}$. The proxy model is architecturally configured to mirror the structural characteristics of either the local prediction heads or the server-side aggregator, ensuring that the proxy learning process reflects the final inference capability. The proxy architecture is specified by the server as part of the calibration protocol, which the server coordinates and delegates to the representative client; only the selected C^* is returned, so no private data leaves the client. Finally, the utility metric $\mathbb{U}(C)$ is computed by first passing $\mathcal{D}_{val}$ through the local encoders to extract representations, which are subsequently evaluated by this trained proxy model.

As C increases, the average intra-cluster distance $\Gamma(C)$ grows, typically leading to a decline in the utility metric $\mathbb{U}(C)$. We seek the "knee" point—the critical threshold where the utility $\mathbb{U}(C)$ begins to drop precipitously relative to the latent space distortion $\Gamma(C)$. Following the same curvature-based selection logic detailed in Appendix A.2, we quantitatively detect this point of maximum curvature using the Kneedle algorithm (Satopää et al., 2011). The complete procedure is summarized in Algorithm 6.

Algorithm 6 Representative Calibration for Optimal Cluster Size

Require: Representative dataset $\mathcal{D}_{rep}$, Candidate search space $\mathcal{CSS}$, User-defined split ratio s

Local Preparation:
Partition $\mathcal{D}_{rep} \to \mathcal{D}_{val}$ with ratio s, and $\mathcal{D}_{train} \leftarrow \mathcal{D}_{rep} \setminus \mathcal{D}_{val}$.
Let $\mathcal{I}_{train}$ be the set of indices for $\mathcal{D}_{train}$.
Optimize local encoder f^{emb} and prediction head f^{head} on $\mathcal{D}_{train}$.
Generate training embeddings $f^{emb}(\mathcal{D}_{train})$ and predictions $\sigma\big(f^{head}(f^{emb}(\mathcal{D}_{train}))\big)$.
Compute the pairwise distance matrix D_{train} of $f^{emb}(\mathcal{D}_{train})$ over $\mathcal{I}_{train}$.
Evaluation Loop:
for each $C \in \mathcal{CSS}$ **do**
    $\mathcal{M}, \mathcal{MS} \leftarrow$ DIVISIVECLUSTERING$(\mathcal{I}_{train}, D_{train}, C)$.
    $\{G_k\}_{k\in\mathcal{M}} \leftarrow \{\mathcal{MS}[\mu] : \mu \in \mathcal{M}\}$. {Convert the returned mapping into membership sets}
    $\Gamma(C) \leftarrow$ Mean intra-cluster distance in $\{G_k\}_{k\in\mathcal{M}}$.
    {Apply local mixup within each cluster to generate synthetic pairs}
    $\tilde{\mathcal{D}} \leftarrow \emptyset$.
    for each cluster G_k **do**
        Generate $|G_k|$ synthetic pairs $\{(\tilde{\mathbf{e}}_i, \tilde{\mathbf{l}}_i)\}_{i=1}^{|G_k|}$ via Dirichlet-weighted interpolation of samples in G_k.
        $\tilde{\mathcal{D}} \leftarrow \tilde{\mathcal{D}} \cup \{(\tilde{\mathbf{e}}_i, \tilde{\mathbf{l}}_i)\}_{i=1}^{|G_k|}$.
    end for
    Train proxy model on the local synthetic set $\tilde{\mathcal{D}}$.
    Evaluate utility $\mathbb{U}(C)$ on $\mathcal{D}_{val}$.
end for
Knee Point Detection:
Apply the Kneedle algorithm to find the point of maximum curvature.
$C^* \leftarrow$ cluster size corresponding to the detected knee point.
return Optimal global cluster size C^*

A.6 SERVER-SIDE TRAINING OF THE ECG-GNN

Algorithm 7 summarizes the server-side training loop of Section 3.5.2.

Algorithm 7 ECG-GNN Training

Require: Synthetic dataset $\tilde{\mathcal{D}}$, Learning rate η, Type-space $\mathcal{T}$
1: Initialize parameters θ_g for ECG-GNN g.
2: **for** each epoch $t = 1, \ldots, T$ **do**
3: Sample a mini-batch $\mathcal{B} \subset \tilde{\mathcal{D}}$.
4: **for** each sample $i \in \mathcal{B}$ **do**
5: Identify the active subset $\mathcal{AC}_i \subseteq \mathcal{CC}_i$.
6: Construct relational graph $\mathcal{R}^{(i)}$ based on $\mathcal{AC}_i$.
7: $\hat{\mathbf{y}}^{(i)} \leftarrow \sigma\left(g(\mathcal{R}^{(i)}; \theta_g)\right)$.
8: **end for**
9: $\mathcal{L}_g \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \ell(\bar{\mathbf{y}}^{(i)}, \hat{\mathbf{y}}^{(i)})$.
10: Update $\theta_g \leftarrow \theta_g - \eta \nabla_{\theta_g} \mathcal{L}_g$.
11: **end for**
12: **return** θ_g

B Details of Experimental Settings

B.1 Multi-News Dataset Adaptation and Analysis

B.1.1 Task Transformation and Preprocessing

To evaluate the proposed framework on textual data, we adapt the Multi-News dataset (Fabbri et al., 2019) for a multi-label classification task. Each sample in the original dataset consists of a set of news articles covering a specific event paired with a human-written summary. The transformation into a classification task is conducted in two distinct stages utilizing `gpt-4o-mini`:

- **Keyword Extraction**: We first extract up to 6 core conceptual keywords from each training summary via the extraction prompt illustrated in Fig. 1. This results in an initial pool of 11,367 unique keywords. To ensure sufficient label density for training, we filter these to the 53 most frequent keywords (frequency > 500) and append a 'None' label to account for samples without relevant keywords, totaling 54 target labels.
- **Label Assignment**: We then utilize the selection prompt (also in Fig. 1) to assign relevant labels to both training and test summaries by matching their content against the established 53-keyword vocabulary.

Extraction Prompt

You are a topic abstraction expert.
Given a news summary, extract generalized, high-level conceptual topics.
Rules:
•Produce up to 6 topic labels for each item.
•Each label must be a general concept (1–3 words), not a specific phrase.
•Avoid numbers, percentages, dates, people, places, organizations, and specific events.
•Convert specific phrases into broader conceptual categories.
•Examples of valid conceptual topics: 'economy', 'labor market', 'public policy', 'healthcare', 'financial markets', 'elections', 'regulation', 'international relations', 'environment'.
•Return ONLY JSON array.

Selection Prompt

You are a topic abstraction expert.
For each news summary, select high-level topic labels ONLY from the given vocabulary.
Rules:
•You MUST choose labels ONLY from the given vocabulary.
•Each returned label MUST exactly match one entry in the vocabulary.
•Do NOT invent new labels.
•Return ONLY a single JSON object mapping sample ID to an array of labels.
•Example: {123: ["economy", "labor"], 124: ["politics"]}
Vocabulary (allowed labels):
[[VOCABULARY LIST]]

Figure 1: Prompts utilized for Multi-News dataset transformation via `gpt-4o-mini`. The extraction prompt identifies high-level conceptual topics to define target classes. The selection prompt labels samples by strictly matching summary content against the 53-keyword vocabulary.

B.1.2 Analysis of Client Data Distribution

We simulate a distributed environment using the training partition of the Multi-News dataset by randomly assigning one article from each sample's set to a client. Since the number of articles per

event varies, if the available articles for a sample are fewer than the total number of clients M, only a subset of clients receives data for that specific sample.

To analyze the resulting feature distribution aggregated across the entire training dataset, we observe token membership within a 10-client environment. To focus on more discriminative features, we filter the vocabulary using Zipf frequency scores (selecting tokens with a score ≥ 2.5) via the Python `wordfreq` library. As illustrated in Fig. 2, a significant majority of tokens ($\approx 35,000$) are shared by all 10 clients, reflecting the linguistic characteristics and high vocabulary overlap inherent in news articles.

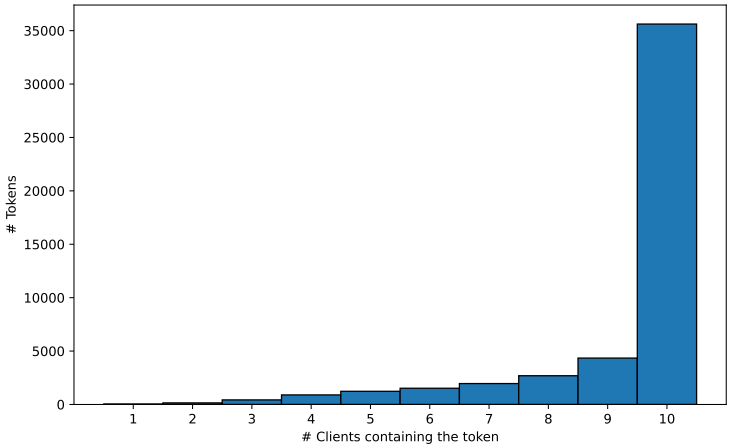


Figure 2: Distribution of token sharing across a 10-client Multi-News environment. The sharp peak at 10 clients reveals a dense overlap of universal tokens, whereas the residual spread indicates the presence of client-unique or domain-specific vocabulary.

B.1.3 Edge Feature Selection

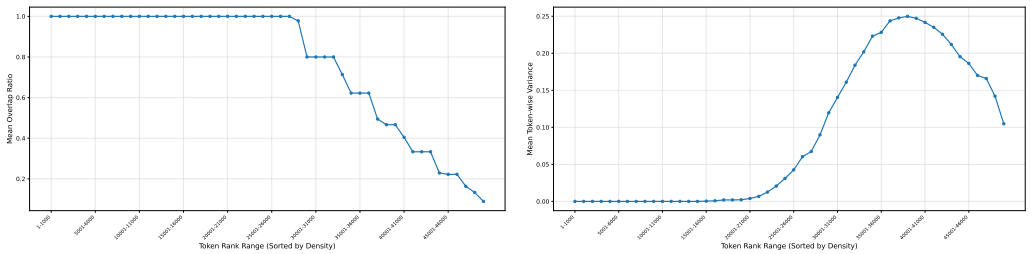
The construction of discriminative edge features based on the statistical properties of the training data requires identifying features that are neither universally redundant nor excessively sparse. Fig. 3 provides the statistical justification for selecting the 35k–45k token rank range for edge feature construction:

- **Mean overlap ratio** (Fig. 3a): High-frequency tokens (ranks $< 28,000$) exhibit an overlap ratio of nearly 1.0. This indicates they are shared by almost every client pair at the individual sample level, offering no informative signal for our ECG-GNN to distinguish between unique client relationships.
- **Mean token-wise variance** (Fig. 3b): By calculating the variance of binary membership for each token across all possible pairs on a per-sample basis, we identify the peak of heterogeneity. The 35k–45k range maximizes the mean token-wise variance (≈ 0.25), providing the most discriminative signals for modeling pairwise client dependencies.

We note that this rank-range selection is a benchmark-construction step: it relies on the cross-client overlap statistics of Fig. 3 computed over the full vocabulary, whereas the deployed server receives token memberships only within the selected range as edge attributes. In practice, the range can be fixed from the Zipf frequencies of a public in-domain corpus without any client data, or the overlap statistics can be aggregated securely with the same THE protocol used for secure distance aggregation in Section 3.3, since only token-level membership counts are required.

B.2 Summary of Experimental Settings

This section provides a consolidated overview of the 12 experimental configurations across three datasets: FashionMNIST, CIFAR-100, and Multi-News. Table 5 summarizes the key parameters for each setting, including the number of clients (M) and the specific architectural strategies employed.



(a) Mean overlap ratio (b) Mean token-wise variance

Figure 3: Sample-level analysis of token overlap characteristics across all client pairs. (a) Mean overlap ratio per 1,000-token chunk, where top-ranked tokens show nearly 100% overlap. (b) Mean of individual token-wise variances.

For settings FM1–FM5 and CI1–CI5, sample distributions and feature/spatial extents are purposefully varied to assess the framework's adaptability to non-uniform federated configurations. In particular, settings FM3–FM5 incorporate a multi-group feature federation structure following the protocol of Zhang et al. (2020b). In these scenarios, features are grouped based on the overlapping patterns of participating clients. For Multi-News (MN1–MN2), the data distribution follows the natural heterogeneity of the dataset. Detailed discussions on the data construction, token selection (e.g., the 35k–45k rank range), and statistical properties for Multi-News are provided in Appendix B.1. Notably, all models across all settings are initialized and trained from scratch.

Table 5: Detailed summary of 12 experimental configurations. In the "Feature/View Info" column, percentages for FM1–FM5 denote the average proportion of the total feature space assigned to each client, while r for CI1–CI5 denotes the side length of the random spatial crop. The strategy column corresponds to the three parameter sharing configurations detailed in Section 3.2.

Setting	Dataset	Clients (M)	Sample %	Feature/View Info	Strategy
FM1	FashionMNIST	50	6%	2.3%	No parameter sharing
FM2	FashionMNIST	50	10%	2.3%	No parameter sharing
FM3	FashionMNIST	50	10%	6.0%	Group-wise selective sharing
FM4	FashionMNIST	50	10%	10.0%	Group-wise selective sharing
FM5	FashionMNIST	50	12%	12.0%	Group-wise selective sharing
CI1	CIFAR-100	10	20%	$r \in [16, 32]$	Global parameter sharing
CI2	CIFAR-100	10	30%	$r \in [16, 32]$	Global parameter sharing
CI3	CIFAR-100	10	40%	$r \in [16, 32]$	Global parameter sharing
CI4	CIFAR-100	8	25%	$r \in [16, 32]$	Global parameter sharing
CI5	CIFAR-100	6	33.3%	$r \in [16, 32]$	Global parameter sharing
MN1	Multi-News	3	-	-	No parameter sharing
MN2	Multi-News	10	-	-	No parameter sharing

C Extended Implementation Details

This appendix provides the full technical specifications and experimental configurations required to reproduce Fed-DREAM.

C.1 Detailed Local Architectures and Modifications

MLP for FashionMNIST: The local encoder for the FashionMNIST dataset is an MLP consisting of Linear-BN-ReLU blocks. To identify the optimal configuration, we conduct a grid search on a single randomly selected client in the FM1 environment. The search space includes: (i) number of layers $\in [3, 10]$, (ii) nodes per layer $\in \{32, 64, 128, 256\}$, and (iii) learning rates $\in \{10^{-2}, 10^{-3}, 10^{-4}\}$ using an Adam optimizer with a 10^{-4} weight decay. Based on a 300-epoch validation run (using

10% of the client's training data), we employ a 4-layer architecture with 128 hidden nodes and set the learning rate to 10^{-3}, as this configuration is identified as the best setup. This configuration is then uniformly applied to all experimental settings (FM1–FM5) to ensure architectural consistency.

Structural Adaptation for Fusion (Settings FM3–FM5): In environments involving multi-group feature federation (FM3–FM5), the encoder architecture is adaptively modified to ensure that all clients produce embeddings of a uniform dimensionality. Once the optimal depth L is determined, we reconfigure the local model as follows:

1. **Group-specific Modules:** Each feature group is processed independently by an encoding module with a depth of $L - 1$.
2. **Final Fusion Layer:** A collaborative fusion layer is integrated as the L-th layer. This layer aggregates the representations from the preceding modules and maps them into a unified latent space.

The primary objective of this fusion layer is to ensure that the representational capacity of the resulting embedding is standardized to the optimal depth L, regardless of the number of feature groups held by a client. By mapping fragmented multi-group views into a uniform feature dimension with consistent information density before they reach the server, we prevent bias in the subsequent node encoding phase and ensure that the ECG-GNN processes balanced representations across all heterogeneous client nodes.

mT5-small for Multi-News: For the Multi-News dataset, we implement a Transformer-based encoder-decoder architecture from scratch following the mT5-small configuration, consisting of 8 encoder and 8 decoder layers. During the local training phase, the model is optimized using the binary cross-entropy for multi-label classification. Specifically, we utilize the full encoder-decoder structure where a single dummy token ($L_{dec} = 1$) is fed into the decoder to activate the cross-attention mechanism, which encourages the encoder to learn more discriminative representations.

The feature extraction for joint inference and relational learning is specifically based on the encoder's output. Let $H \in \mathbb{R}^{T \times d}$ be the hidden states from the L-th encoder layer (where $L = 6$ for Multi-News), with $T = 512$ denoting the maximum sequence length and d representing the hidden dimension size. To condense these states into a sequence-independent latent vector $\mathbf{h}_{pooled}$, we apply the following masked mean pooling strategy:

$$\mathbf{h}_{pooled} = \frac{\sum_{t=1}^{T}(H_t \odot PM_t)}{\sum_{t=1}^{T} PM_t + 10^{-6}}, \tag{9}$$

where $PM \in \{0, 1\}^T$ is a binary padding mask indicating valid tokens, PM_t denotes the t-th element of PM, and 10^{-6} is a stability constant to prevent division by zero.

C.2 Summary of Local Training Hyperparameters

Table 6 summarizes the local training configurations used for each dataset across all experimental environments. For configurations that involve parameter sharing, clients perform one local epoch per communication round.

Table 6: Detailed local training hyperparameters.

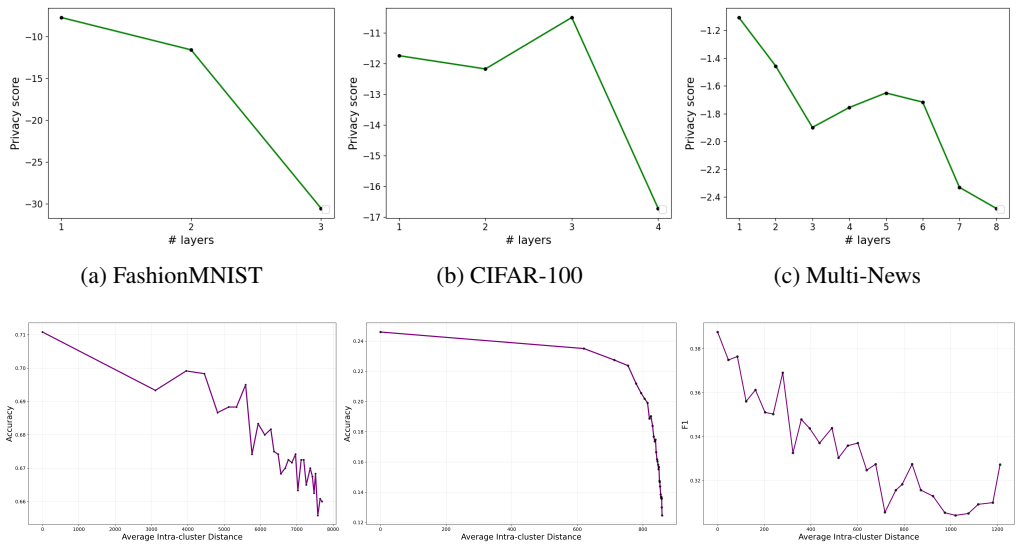
Parameter	FashionMNIST	CIFAR-100	Multi-News
Encoder Architecture	4-layer MLP	ResNet-18	mT5-small
Optimizer	Adam	SGD	AdamW
Learning Rate (Initial)	10^{-3}	0.1	10^{-4}
LR Schedule	-	Step [50, 75] ($\times 0.1$)	Cosine (0.06 ratio)
Weight Decay	10^{-4}	10^{-4}	0
Batch Size	128	128	16
Total Local Epochs	300	100	20
Dropout	-	-	0.1

C.3 Selection Rationale for Structural and Privacy Parameters

To optimize the trade-off between privacy and utility, we pre-determine the key hyperparameters—encoder depth (L) and minimum cluster size (C)—prior to the collaborative training phase. These parameters are calibrated using a single randomly selected client from the primary experimental configurations (FM1, CI1, and MN1). For the multi-sample representation-level mixup, the Dirichlet parameter α is set to 1.0 for all datasets. This configuration corresponds to a uniform distribution over the simplex (Ng et al., 2011), serving as a non-informative prior for privacy-preserving interpolation.

First, regarding the selection of encoder depth (L), we employ the privacy scoring protocol (s_m^j) formulated via the bi-level optimization problem defined in Appendix A.2. For this process, we set the sensitivity δ to 10,000 and the Laplace scale to 0.0001. Additionally, clipping values for inner maximization and outer minimization are fixed at 1 and 10^{-5}, respectively, to ensure numerical stability.

Subsequently, the minimum cluster size (C) is determined using task-specific proxy models to evaluate the impact of clustering on representation utility. For FashionMNIST and CIFAR-100, we utilize the original local classifiers (the layers succeeding the encoder) as proxies. For Multi-News, as the mT5-small encoder produces a unified 1D embedding, we employ a 3-block MLP as a proxy. This architecture is intentionally designed to mirror the 3-block configuration of the downstream ECG-GNN (consisting of node encoding, relational message passing, and graph-level readout and prediction module), ensuring structural compatibility during the parameter search. Based on these experimental setups, the optimal values were determined as $L = 2, 3, 6$ and $C = 4, 5, 3$ for FashionMNIST, CIFAR-100, and Multi-News, respectively, as illustrated in Fig. 4.



(a) FashionMNIST (b) CIFAR-100 (c) Multi-News

(d) FashionMNIST (e) CIFAR-100 (f) Multi-News

Figure 4: Selection of optimal encoder depth (L) and minimum cluster size (C). The top row (a–c) plots privacy score vs. layer depth, and the bottom row (d–f) depicts performance and average intra-cluster distance as C is incremented by 1 unit. For both parameters, the optimal configurations are identified at the "knee" points, which signify the critical thresholds for balancing privacy-preserving sensitivity (top) and the utility-distortion trade-off in clustering (bottom).

C.4 Server-side Architecture and Training Protocols

Architectural Specifications. For simplicity and to ensure a consistent evaluation across diverse datasets, the server-side ECG-GNN is configured with a standardized structure. Specifically, we employ a three-stage GNN architecture consisting of (i) a node encoding module, (ii) a single-layer message passing module ($L_g = 1$), and (iii) a graph-level readout and prediction module. To minimize task-specific hyperparameter tuning, the hidden dimensions for both the message and gate components

are fixed at 256, while a 0.3 dropout rate is applied only to the message component. Specifically, both the message generation function $\phi^{(l)}$ and the gating function $\psi^{(l)}$ follow a linear-ReLU-linear structure. The node encoding module serves as a bridge to align various local embeddings with this unified 256-dimensional latent space. Table 7 summarizes the specific architectural configurations of the node encoding module.

Table 7: Architectural details of node encoding modules.

Dataset	**Node Encoding Architecture**
FashionMNIST	Linear(units=256) $\rightarrow$ ReLU
CIFAR-100	Block 1: [Conv(channels=128, kernel size=3×3, padding=1) + Batch Normalization + ReLU + Dropout(0.3)] $\rightarrow$ Block 2: [Conv(channels=256, kernel size=3×3, padding=1) + Batch Normalization + ReLU + Dropout(0.3)] $\rightarrow$ Global Average Pooling
Multi-News	Linear(units=256) $\rightarrow$ ReLU

Server Training Protocols and Hyperparameters. The ECG-GNN is trained using the Adam optimizer with a batch size of 128 and an initial learning rate of 0.001. The learning rate is decayed by a factor of 0.1 at specific milestones to ensure convergence. Table 8 details the training duration and associated hyperparameters for each dataset.

Table 8: Hyperparameters for server-side ECG-GNN training.

Hyperparameter	**FashionMNIST**	**CIFAR-100**	**Multi-News**
Optimizer	Adam	Adam	Adam
Initial Learning Rate	0.001	0.001	0.001
Learning Rate Decay	$\times 0.1$	$\times 0.1$	$\times 0.1$
Decay Milestones (Epoch)	[10]	[10, 30]	[10]
Total Epochs	20	50	20
Batch Size	128	128	128

C.5 Detailed Benchmark Implementations

Rather than focusing on the training of an independent global server model, the compared benchmarks prioritize the enhancement or direct utilization of individual local models through collaborative communication. To ensure a rigorous comparison, we standardize both the training configurations (e.g., local epochs and communication rounds) and the model architectures of the baselines to match our local setup. Unless explicitly stated below, all baseline hyperparameters and structural components are identical to those of our proposed framework. Additional information regarding training-specific settings and structural modifications is as follows:

- **FedLabSync & FedLabSync-WA**: For both versions, we set the prediction weight hyperparameter to 0.5 to equally weight the contributions from target and neighboring clients as suggested by Llasag Rosero et al. (2025). FedLabSync-WA is omitted in FM1, FM2, and Multi-News: in FM1 and FM2 the overlapping feature group sizes are too small to support stable weight averaging, and on Multi-News no parameters are shared; since Fed-DREAM also uses no parameter sharing in these settings, the comparison there is between two methods without parameter sharing.
- **HyFEM**: We conduct a grid search for the balancing hyperparameter, which controls the trade-off between local task-specific optimization and collaborative regularization, within the range [0.1, 1.0] with increments of 0.1. Since the convergence of HyFEM is established by viewing it as block-coordinate gradient descent, we retain plain SGD as the local solver so that the analyzed update rule is preserved, with the initial learning rate searched within the range $\{10^{-5}, 10^{-4}, \ldots, 1.0\}$. For both hyperparameters, the values that yielded the highest performance in the primary configurations (FM1 and CI1) are selected

and consistently applied across all other experimental settings. Specifically, the optimal balancing hyperparameter is set to 0.1 for FashionMNIST and 1.0 for CIFAR-100, while the optimal initial learning rate is 0.001 for FashionMNIST and 0.1 for CIFAR-100.

- **FedHD**: The original single-layer encoder design is generalized to a flexible multi-layer structure to match our local model capacity. Regarding hyperparameter tuning, only the initial learning rate is searched for FedHD, following the same search space and selection criteria defined for HyFEM. Since FedHD's gradient-tracking correction is defined on raw stochastic gradients, we retain SGD as in the original algorithm; the optimal initial learning rates are identified as 1.0 for FashionMNIST and 0.01 for CIFAR-100.

D Detailed Performance Comparison with Benchmark Algorithms

This appendix provides comprehensive experimental results, with each table cell reporting the mean performance (top) and its corresponding standard deviation (bottom). For FashionMNIST and CIFAR-100, results are averaged over five independent trials using Accuracy, as detailed in Table 9. In contrast, the Multi-News results, presented in Table 10, are obtained from three independent trials. Given the multi-label nature of the Multi-News task, we employ Micro-F1 score, mAP, and Hamming Loss for evaluation.

Results on FashionMNIST and CIFAR-100. Table 9 presents the detailed classification performance for the FashionMNIST and CIFAR-100 datasets across different participation rates ($P \in \{1.0, 0.3\}$). Notably, the standard deviations for CIFAR-100 are relatively higher, as the inherent complexity of the dataset exacerbates the variance introduced by the random assignment of sample and feature indices across independent runs. For $P = 0.3$, the number of training epochs was tripled for all methods—to compensate for the reduced number of participating clients per round, so that the total amount of training remains comparable to $P = 1.0$.

Table 9: Full classification results on FashionMNIST and CIFAR-100. Each cell reports mean $\pm$ standard deviation. The *p*-val columns report the p-values of paired one-sided t-tests between Fed-DREAM and each benchmark, pooling all settings per dataset.

P	Method	FashionMNIST (Tabular)						CIFAR-100 (Image)					
		FM1	FM2	FM3	FM4	FM5	*p*-val	CI1	CI2	CI3	CI4	CI5	*p*-val
1.0	FedHD	0.5826 ±0.0165	0.6606 ±0.0074	0.7125 ±0.0173	0.7804 ±0.0034	0.7885 ±0.0045	<0.001	0.3210 ±0.0269	0.3615 ±0.0237	0.3953 ±0.0128	0.3123 ±0.0147	0.3002 ±0.0302	<0.001
	HyFEM	0.7308 ±0.0040	0.7829 ±0.0043	0.8292 ±0.0013	0.8324 ±0.0189	0.8539 ±0.0019	<0.001	0.3068 ±0.0055	0.4067 ±0.0120	0.4779 ±0.0103	0.3285 ±0.0107	0.3816 ±0.0173	<0.001
	FedLabSync	0.7492 ±0.0065	0.7938 ±0.0043	0.8343 ±0.0010	0.8466 ±0.0021	0.8565 ±0.0023	<0.001	0.2599 ±0.0209	0.3593 ±0.0202	0.3998 ±0.0091	0.3053 ±0.0152	0.3578 ±0.0176	<0.001
	FedLabSync-WA	- -	- -	0.8326 ±0.0013	0.8499 ±0.0012	0.8605 ±0.0024	<0.001	0.4081 ±0.0223	0.4793 ±0.0260	0.5338 ±0.0200	0.3897 ±0.0223	0.4285 ±0.0208	<0.001
	Fed-DREAM	**0.7493 ±0.0062**	**0.7946 ±0.0047**	**0.8371 ±0.0018**	**0.8572 ±0.0013**	**0.8659 ±0.0019**	-	**0.4389 ±0.0239**	**0.4952 ±0.0293**	**0.5737 ±0.0143**	**0.4213 ±0.0237**	**0.4513 ±0.0288**	-
0.3	FedHD	0.5854 ±0.0156	0.6720 ±0.0043	0.7721 ±0.0047	0.7922 ±0.0039	0.7927 ±0.0072	<0.001	0.3321 ±0.0165	0.3848 ±0.0242	0.4441 ±0.0229	0.3791 ±0.0213	0.2738 ±0.0402	<0.001
	HyFEM	0.7365 ±0.0057	0.7825 ±0.0035	0.8297 ±0.0015	0.8453 ±0.0029	0.8514 ±0.0020	<0.001	0.3109 ±0.0037	0.3867 ±0.0050	0.4648 ±0.0097	0.3687 ±0.0136	0.3241 ±0.0110	<0.001
	FedLabSync	**0.7511 ±0.0048**	**0.7946 ±0.0039**	0.8322 ±0.0011	0.8471 ±0.0012	0.8555 ±0.0010	0.010	0.2316 ±0.0270	0.3553 ±0.0244	0.3899 ±0.0410	0.3168 ±0.0129	0.3053 ±0.0314	<0.001
	FedLabSync-WA	- -	- -	0.8330 ±0.0012	0.8500 ±0.0034	0.8613 ±0.0010	<0.001	0.3642 ±0.0202	0.4494 ±0.0249	0.5060 ±0.0237	0.3733 ±0.0113	0.4479 ±0.0168	<0.001
	Fed-DREAM	0.7475 ±0.0051	0.7936 ±0.0033	**0.8345 ±0.0015**	**0.8556 ±0.0017**	**0.8668 ±0.0034**	-	**0.4479 ±0.0077**	**0.5280 ±0.0206**	**0.5897 ±0.0135**	**0.4303 ±0.0362**	**0.4734 ±0.0363**	-

Results on Multi-News. Table 10 presents the full experimental results for Multi-News ($P = 1.0$), obtained from three independent trials.

Table 10: Full classification results on Multi-News. Each cell reports mean ± standard deviation. For Hamming Loss, lower is better (↓); for others, higher is better (↑). The *p-value* row reports paired one-sided t-tests against FedLabSync per metric, pooling all settings.

P	Method	MN1 ($M = 3$)			MN2 ($M = 10$)		
		Micro-F1 ↑	mAP ↑	Hamming ↓	Micro-F1 ↑	mAP ↑	Hamming ↓
1.0	FedLabSync	0.4176 ±0.0039	0.3251 ±0.0053	0.0416 ±0.0017	0.3787 ±0.0013	0.2707 ±0.0040	0.0385 ±0.0006
	Fed-DREAM	**0.4358 ±0.0005**	**0.3478 ±0.0018**	**0.0312 ±0.0009**	**0.3850 ±0.0016**	**0.3050 ±0.0016**	**0.0300 ±0.0006**
	p-value	Micro-F1: 0.018		mAP: 0.016		Hamming: 0.006	

Definition of the Reference Conditions. *Local-only* trains, on each client, the same local encoder and prediction head as Fed-DREAM on that client's own samples and features, without any communication; at test time each sample is predicted independently by every client holding it from that client's partial features, and the reported accuracy averages, for each test sample, the correctness of its holders' predictions and then averages over all test samples (no predictions are ensembled; for Multi-News, Micro-F1 and Hamming loss are accumulated analogously with each sample–holder pair weighted by the inverse number of holders, and mAP is averaged over clients). It measures what a client can achieve alone, so a collaborative method that falls below it indicates negative transfer from collaboration. *Oracle feature union* instead assembles, for every sample, the features held by all of its holders in the original input space, filling unobserved entries with zeros (the union of the holders' feature indices over the 784-dimensional vector for FashionMNIST, and the holders' crop regions placed at their original coordinates on the 32×32 canvas without resizing for CIFAR-100), and trains a single model with the same architecture as the local model but full-dimensional input (the MLP for FashionMNIST, ResNet-18 for CIFAR-100) centrally on all N samples with the local training settings of Table 6; test inputs are assembled in the same way. It represents the naive centralized alternative that pools raw features without modeling which entries are missing, and is omitted for Multi-News for the reason given in Section 4.3. The *Raw-embedding* reference of Section 4.5 is defined there.

Results for the Reference Conditions. Table 11 reports the full results for the reference conditions of Section 4.3 – *Local-only* and *Oracle feature union* – and the *Raw-embedding* reference of Section 4.5, including standard deviations. Multi-News results are reported separately in Table 12. FedHD falls below Local-only in four of the five FashionMNIST settings and in CI5, while exceeding it in the other four CIFAR-100 settings; we attribute this to its gradient-tracking correction, which injects global gradient information into local updates and becomes misaligned with local objectives when clients share only 2.3–12% of the features among $M = 50$ clients, whereas the 16–32 pixel crops of CIFAR-100 overlap substantially.

E Comprehensive Component Analysis

This section provides an in-depth analysis of individual framework components. All experiments are conducted under the $P = 1.0$ setting.

E.1 Ablation Study

To validate the contribution of each design element in Fed-DREAM, we evaluate the following ablated versions:

Table 11: Full results for the reference conditions on FashionMNIST and CIFAR-100.

	FM1	**FM2**	**FM3**	**FM4**	**FM5**	**CI1**	**CI2**	**CI3**	**CI4**	**CI5**
Local-only	0.6501 ±0.0018	0.6615 ±0.0029	0.7532 ±0.0018	0.7794 ±0.0023	0.7942 ±0.0013	0.2533 ±0.0067	0.3034 ±0.0103	0.3411 ±0.0146	0.2752 ±0.0164	0.3212 ±0.0122
Oracle feature union	0.7070 ±0.0053	0.7534 ±0.0047	0.8062 ±0.0047	0.8269 ±0.0043	0.8405 ±0.0027	0.4649 ±0.0290	0.5055 ±0.0331	0.5592 ±0.0217	0.4491 ±0.0327	0.4762 ±0.0321
Raw-embedding (P=1.0)	0.7335 ±0.0064	0.7795 ±0.0045	0.8317 ±0.0023	0.8512 ±0.0018	0.8624 ±0.0021	0.4430 ±0.0075	0.5103 ±0.0236	0.5770 ±0.0132	0.4426 ±0.0188	0.4804 ±0.0250
Raw-embedding (P=0.3)	0.7437 ±0.0055	0.7941 ±0.0037	0.8397 ±0.0023	0.8576 ±0.0016	0.8686 ±0.0024	0.4948 ±0.0195	0.5544 ±0.0218	0.6115 ±0.0161	0.4728 ±0.0257	0.5077 ±0.0214

Table 12: Full results for the reference conditions on Multi-News (P=1.0).

	MN1 ($M = 3$)			**MN2** ($M = 10$)		
	Micro-F1 ↑	**mAP** ↑	**Hamming** ↓	**Micro-F1** ↑	**mAP** ↑	**Hamming** ↓
Local-only	0.3818 ±0.0004	0.2825 ±0.0018	0.0430 ±0.0000	0.3168 ±0.0016	0.2358 ±0.0033	0.0466 ±0.0001
Raw-embedding	0.4345 ±0.0042	0.3472 ±0.0005	0.0392 ±0.0003	0.3892 ±0.0294	0.2955 ±0.0176	0.0384 ±0.0024

- *w/o edge feature*: To eliminate the influence of relational context, the edge feature $\mathbf{a}_{mj}$ is replaced with a zero vector for both the message generation and gating functions within the message passing module.
- *w/o gate*: The gating mechanism, which modulates message weights based on relational importance, is disabled, assigning a constant weight of 1 to all incoming messages.
- *Mean-pool MLP*: The message passing and gating of the ECG-GNN are removed while the node encoding module is retained; the encoded client embeddings of each sample are mean-pooled and fed to an MLP that produces the prediction. All hyperparameters and data partitions are kept identical to those of the ECG-GNN. This variant serves as a control for server-side fusion without any relational structure, isolating the contribution of relational learning.
- *w/o node connection*: Logit aggregation across clients from Section 3.4 is omitted. Individual nodes learn independently, and their outputs are averaged only during the inference phase.
- *w/o mixup*: The multi-sample representation-level mixup is omitted. Instead, only the embeddings and corresponding logits of the cluster centroids are utilized as training data.

The results on FashionMNIST (Table 13) and CIFAR-100 (Table 14) demonstrate that every component significantly contributes to the final performance. Notably, the relational aggregation via ECG-GNN and mixup are identified as the most dominant factors. While the performance degradation in the *w/o mixup* setting is somewhat expected due to the reduced sample size for ECG-GNN training when utilizing only cluster centroids, our proposed multi-sample representation-level mixup framework ensures robust performance while retaining the intended obfuscation of shared representations. Furthermore, the introduction of edge feature and gated message also yields meaningful improvements, even if their relative impact is smaller compared to the aforementioned two components. While the standard deviations for CIFAR-100 are relatively higher because the sample and feature indices assigned to clients vary across independent runs, paired one-sided t-tests across 25 independent runs (5 settings $\times$ 5 trials) conducted for each dataset confirm that the inclusion of these two elements provides statistically significant performance gains. P-values are reported in the Avg. column to provide rigorous verification, particularly for cases with relatively small performance

margins. Relative to Mean-pool MLP, Fed-DREAM is higher in all ten settings, by 2.2pp on average and most markedly on the low-overlap FM1–FM2, with the pooled tests significant on both datasets (Tables 13 and 14). Comparing the three server-side designs clarifies where the gain comes from. *w/o node connection* averages the clients' independent outputs (late fusion); *Mean-pool MLP* fuses their embeddings before prediction but ignores which client contributed what (structure-free early fusion); Fed-DREAM fuses embeddings while conditioning on inter-client relations (relational early fusion). On CIFAR-100 and FM3–FM5, accuracy rises in this order (e.g., $0.4191 \rightarrow 0.4638 \rightarrow 0.4761$ on the CIFAR-100 average), so fusing embeddings helps and relational modeling adds further. On FM1–FM2, where each client holds only 2.3% of the features, plain embedding fusion is actually worse than late fusion (e.g., 0.6641 vs. 0.6940 on FM1): averaging embeddings from clients that see almost disjoint features blurs them, and only the relational structure of ECG-GNN turns early fusion into a gain (0.7493 on FM1).

Table 13: Ablation study on FashionMNIST. Each cell reports mean ± standard deviation, and p-values from paired one-sided t-tests are provided in parentheses in the Avg. column.

Setting	**FM1**	**FM2**	**FM3**	**FM4**	**FM5**	**Avg.**
Fed-DREAM	**0.7493** **±0.0062**	**0.7946** **±0.0047**	0.8371 ±0.0018	**0.8572** **±0.0013**	**0.8659** **±0.0019**	**0.8208**
w/o edge feature	0.7459 ±0.0061	0.7911 ±0.0043	0.8366 ±0.0019	0.8567 ±0.0012	0.8652 ±0.0008	0.8191 (0.001)
w/o gate	0.7487 ±0.0063	0.7935 ±0.0032	0.8356 ±0.0038	0.8563 ±0.0012	0.8655 ±0.0019	0.8199 (0.013)
Mean-pool MLP	0.6641 ±0.0096	0.7220 ±0.0048	0.8356 ±0.0026	0.8549 ±0.0010	0.8649 ±0.0006	0.7883 (<0.001)
w/o connection	0.6940 ±0.0056	0.7402 ±0.0082	0.8262 ±0.0025	0.8482 ±0.0021	0.8569 ±0.0020	0.7931 (<0.001)
w/o mixup	0.7447 ±0.0063	0.7908 ±0.0042	**0.8373** **±0.0025**	0.8566 ±0.0008	0.8656 ±0.0013	0.8190 (0.004)

Table 14: Ablation study on CIFAR-100. Each cell reports mean ± standard deviation, and p-values from paired one-sided t-tests are provided in parentheses in the Avg. column.

Setting	**CI1**	**CI2**	**CI3**	**CI4**	**CI5**	**Avg.**
Fed-DREAM	**0.4389** **±0.0239**	0.4952 ±0.0293	**0.5737** **±0.0143**	**0.4213** **±0.0237**	**0.4513** **±0.0288**	**0.4761**
w/o edge feature	0.4310 ±0.0220	**0.4988** **±0.0230**	0.5681 ±0.0146	0.4171 ±0.0225	0.4509 ±0.0295	0.4732 (0.018)
w/o gate	0.4308 ±0.0243	0.4939 ±0.0238	0.5661 ±0.0192	0.4195 ±0.0195	0.4502 ±0.0293	0.4721 (0.002)
Mean-pool MLP	0.4264 ±0.0264	0.4847 ±0.0254	0.5532 ±0.0167	0.4078 ±0.0227	0.4467 ±0.0261	0.4638 (<0.001)
w/o connection	0.3898 ±0.0247	0.4338 ±0.0281	0.4969 ±0.0165	0.4028 ±0.0280	0.3723 ±0.0192	0.4191 (<0.001)
w/o mixup	0.3829 ±0.0225	0.4532 ±0.0256	0.5248 ±0.0187	0.3610 ±0.0208	0.3921 ±0.0275	0.4228 (<0.001)

E.2 Comparative Analysis of Gating Mechanisms and Encoding Strategies

We evaluate the performance impact of different gating functions in the message passing module and the selection of node encoding modules. As summarized in Tables 15 and 16, the Softmax gate (where the sum of neighbor weights is normalized to 1) outperforms the Sigmoid gate (which scales each neighbor's weight between 0 and 1), particularly in the data-constrained CIFAR-100 environments. In contrast, the performance gap between these mechanisms is negligible for FashionMNIST.

Regarding the selection of node encoding modules, the optimal strategy depends on the underlying parameter sharing policy. For FashionMNIST, where local models are not fully synchronized through parameter sharing, the client-specific node encoding strategy yields superior results, notably in the fully isolated FM1 and FM2 settings (see Table 17). This confirms the necessity of node-specific modules for aligning fragmented and unsynchronized feature spaces. Conversely, for CIFAR-100 (Table 18), which utilizes global parameter sharing for local model training, a shared node encoding module achieves better performance by exploiting synchronized representations.

Table 15: Impact of different gating functions on FashionMNIST. Each cell reports mean ± standard deviation.

Setting	**FM1**	**FM2**	**FM3**	**FM4**	**FM5**	**Avg.**
Sigmoid gate	0.7481 ±0.0077	0.7942 ±0.0035	0.8370 ±0.0018	**0.8578 ±0.0013**	**0.8663 ±0.0017**	0.8207
Softmax gate	**0.7493 ±0.0062**	**0.7946 ±0.0047**	**0.8371 ±0.0018**	0.8572 ±0.0013	0.8659 ±0.0019	**0.8208**

Table 16: Impact of different gating functions on CIFAR-100. Each cell reports mean ± standard deviation.

Setting	**CI1**	**CI2**	**CI3**	**CI4**	**CI5**	**Avg.**
Sigmoid gate	0.4310 ±0.0254	0.4949 ±0.0248	0.5658 ±0.0160	0.4153 ±0.0232	0.4475 ±0.0321	0.4709
Softmax gate	**0.4389 ±0.0239**	**0.4952 ±0.0293**	**0.5737 ±0.0143**	**0.4213 ±0.0237**	**0.4513 ±0.0288**	**0.4761**

Table 17: Effect of node encoding module selection on FashionMNIST. Each cell reports mean ± standard deviation.

Setting	**FM1**	**FM2**	**FM3**	**FM4**	**FM5**	**Avg.**
Shared node encoding	0.6584 ±0.0089	0.7140 ±0.0050	0.8335 ±0.0024	0.8541 ±0.0016	0.8637 ±0.0009	0.7847
Client-specific node encoding	**0.7493 ±0.0062**	**0.7946 ±0.0047**	**0.8371 ±0.0018**	**0.8572 ±0.0013**	**0.8659 ±0.0019**	**0.8208**

F Extended Privacy Evaluation

This appendix provides the full experimental protocol and complete results for the empirical privacy audit summarized in Section 4.5.

Table 18: Effect of node encoding module selection on CIFAR-100. Each cell reports mean ± standard deviation.

Setting	CI1	CI2	CI3	CI4	CI5	Avg.
Shared node encoding	**0.4389** **±0.0239**	0.4952 ±0.0293	**0.5737** **±0.0143**	**0.4213** **±0.0237**	**0.4513** **±0.0288**	**0.4761**
Client-specific encoder	0.4166 ±0.0181	**0.4956** **±0.0240**	0.5688 ±0.0119	0.4122 ±0.0154	0.4424 ±0.0332	0.4671

F.1 Membership Inference: Full Protocol and Extended Results

Adversary knowledge and membership definition. The adversary has query access to the deployed system and observes its per-sample loss. Membership is defined at the system level: the member set is $\bigcup_m \mathcal{N}_m$ – every original training sample held by at least one client (60,000 for FM and 50,000 for CI, with full coverage) – and the non-member set is the entire held-out test split (10,000 samples). No subsampling to equal set sizes is performed: AUC is rank-based and invariant to the member/non-member ratio, and TPR@FPR is defined on the non-member axis alone. The audit targets the deployed inference path; the local encoders remain on client devices and are not exposed to third parties, hence attacks against them fall outside this threat model.

Query rule and scores. A query for sample n is served by its holder set $\mathcal{CC}_n = \{m : n \in \mathcal{N}_m\}$. Following the loss-based attack of Yeom et al. (2018), the membership score is the negated per-sample cross-entropy loss under which each system was trained; sweeping the decision threshold over all scores traces out the full ROC curve. The attack respects the different deployment forms of the two systems: Fed-DREAM exposes a single inference path – the graph over $\mathcal{CC}_n$ is passed once through the server-side ECG-GNN, yielding one loss per query – whereas FedLabSync deploys M local models, so each client $m \in \mathcal{CC}_n$ evaluates its partial view $\mathbf{x}_{m,n}$, producing per-client losses $\{\ell_m\}_{m\in\mathcal{CC}_n}$, aggregated into the membership score $-\operatorname{mean}_{m\in\mathcal{CC}_n} \ell_m$. Consistent with the main text, the attacked baseline in each setting is the highest-utility benchmark of Table 1: FedLabSync-WA, or FedLabSync on FM1. Table 19 lists the complete results, including TPR@0.1%FPR, which is omitted from Table 4 because only ten non-member false positives are admissible at that operating point.

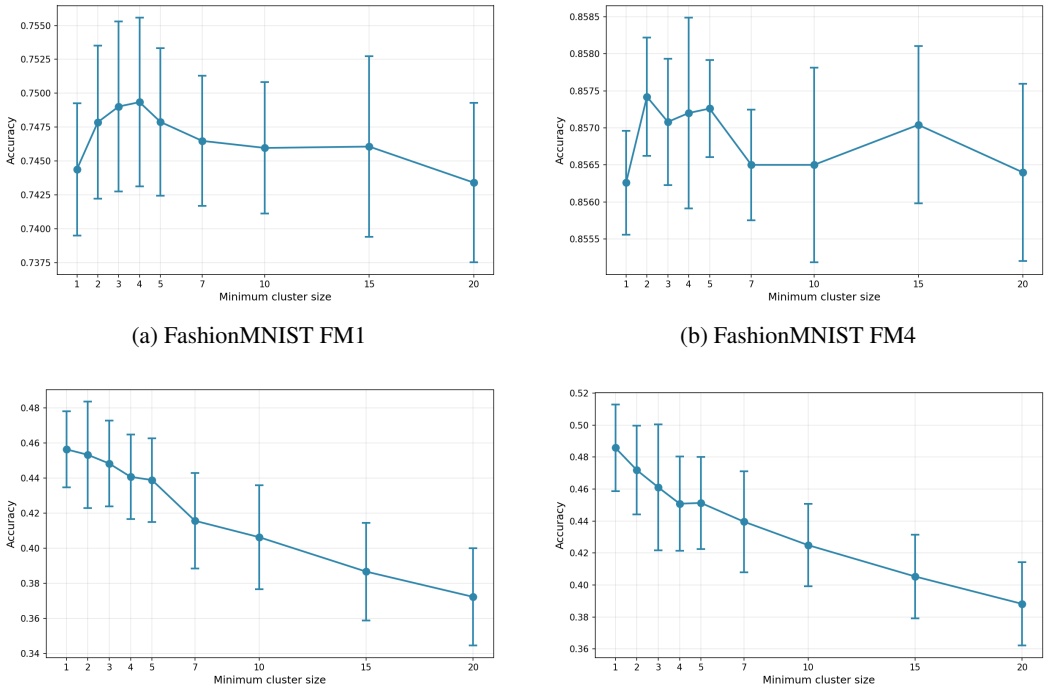
Table 19: Full membership inference results (mean ± std) of Fed-DREAM, using the deployed cluster sizes, and the highest-utility benchmark in each setting (FedLabSync on FM1, FedLabSync-WA otherwise): AUC, TPR@1%FPR, and TPR@0.1%FPR. Random-guess levels are 0.5, 0.01, and 0.001, respectively.

Setting	Method	AUC	TPR@1%FPR	TPR@0.1%FPR
FM1	**Fed-DREAM**	**0.6145 ± 0.0018**	0.0112 ± 0.0009	**0.0011 ± 0.0004**
	FedLabSync	0.6449 ± 0.0020	**0.0107 ± 0.0016**	0.0014 ± 0.0005
FM4	**Fed-DREAM**	**0.5768 ± 0.0016**	**0.0099 ± 0.0009**	**0.0010 ± 0.0001**
	FedLabSync-WA	0.6302 ± 0.0029	0.0114 ± 0.0003	0.0011 ± 0.0000
CI1	**Fed-DREAM**	**0.6094 ± 0.0039**	**0.0151 ± 0.0016**	0.0019 ± 0.0005
	FedLabSync-WA	0.8490 ± 0.0060	0.0191 ± 0.0015	**0.0014 ± 0.0004**
CI5	**Fed-DREAM**	**0.6060 ± 0.0072**	**0.0157 ± 0.0025**	**0.0014 ± 0.0002**
	FedLabSync-WA	0.8776 ± 0.0148	0.0271 ± 0.0025	0.0027 ± 0.0003

F.2 Privacy–Utility Sensitivity to the Minimum Cluster Size

Utility. The minimum cluster size (C) is a pivotal hyperparameter that manages the trade-off between data privacy and model utility. Figure 5 illustrates the performance trends as C is varied from 1 to 20. In the CIFAR-100 configurations, accuracy tends to decline as C increases due to the data-constrained nature of the task, reinforcing the rationale for the calibration guidelines proposed in Appendix A.5.

In contrast, for FashionMNIST, the performance remains remarkably stable even at $C = 20$. This indicates that for datasets where generalization is more attainable, larger cluster sizes can be adopted to further obfuscate individual sample contributions, thereby enhancing privacy protection without incurring significant utility costs, potentially exceeding our automated selection protocol.



(a) FashionMNIST FM1

(b) FashionMNIST FM4

(c) CIFAR-100 CI1

(d) CIFAR-100 CI5

Figure 5: Sensitivity analysis of the minimum cluster size (C) on classification accuracy (mean $\pm$ standard deviation) across different settings. Error bars represent the standard deviation across trials.

Membership leakage. Figure 6 shows the corresponding dependence of leakage on C: Fed-DREAM's AUC decreases monotonically with C in all four settings, so C acts as the privacy knob of the framework, and on CIFAR-100 Fed-DREAM lies far below the FedLabSync-WA baseline for every C, including the degenerate unclustered condition C=1. The low-FPR operating points, in contrast, are at the random-guess level already without any mixing and are essentially insensitive to C: at C=1, Fed-DREAM's TPR@1%FPR is 1.20%/1.00%/1.33%/1.24% on FM1/FM4/CI1/CI5 against the 1% baseline and its TPR@0.1%FPR is 0.12%/0.08%/0.13%/0.13% against the 0.1% baseline, and across the entire C sweep the values stay within 0.9–1.7% and 0.07–0.20%, respectively. We note transparently that on FM1 the C=1 condition (AUC 0.6555) slightly exceeds the FedLabSync baseline (0.6449); Fed-DREAM falls below the baseline from $C \geq 2$ onward, and the deployed configuration uses C=4.

G System Efficiency

This appendix profiles the system costs specific to Fed-DREAM: the communication volume and THE workload of Stage 1 of the federated divisive clustering (Section 3.3), the upload volume of the synthetic embeddings and logits produced by the representation-level mixup, and stage-wise wall-clock times. All profiles use P=1.0 and a single run with a fixed seed per setting. The accuracy experiments in this paper were executed with plaintext aggregation; since THE recovers the exact aggregated sums, all reported accuracy results are identical under encryption, and the cryptographic costs below are estimates obtained by multiplying measured single-operation times (Table 22) by analytically counted operation numbers. Communication for local parameter sharing is

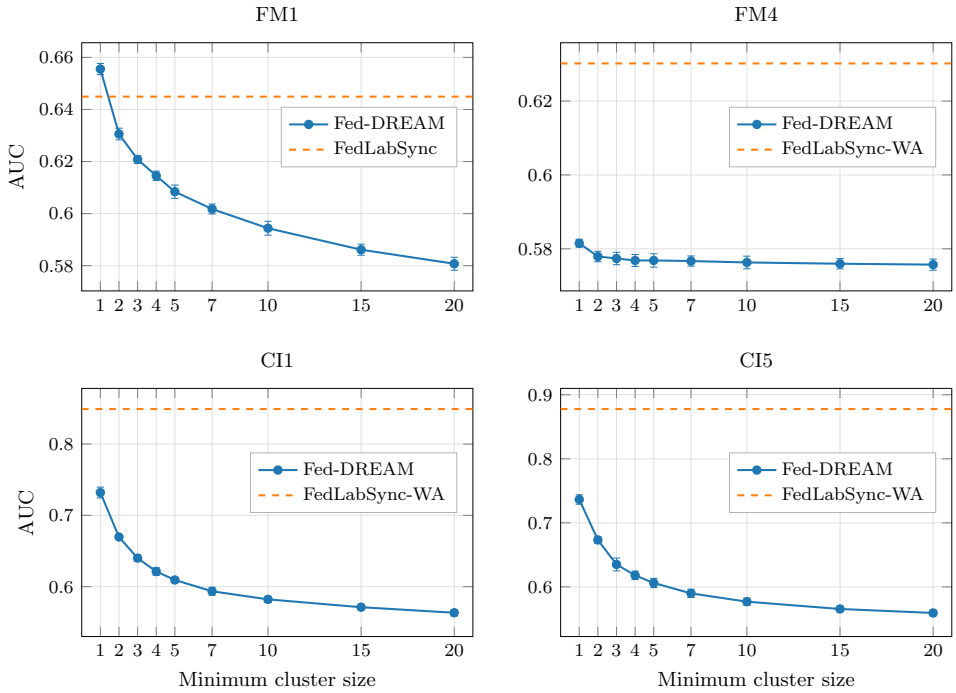


Figure 6: Membership inference AUC of Fed-DREAM versus the minimum cluster size C (mean $\pm$ standard deviation), with the per-setting benchmark shown as a dashed horizontal line.

common to federated baselines and is therefore excluded; the totals below isolate the costs specific to Fed-DREAM's clustering and representation-sharing stages.

Communication and THE workload. Table 20 reports, per setting, the number of encrypted items $\Sigma|V| = \sum_m \binom{|\mathcal{N}_m|}{2}$ uploaded by clients, the number of aggregated entries N_{pair} decrypted by the server, the resulting Stage-1 traffic, the plaintext mixup transfer, and the estimated THE time under per-client parallel encryption with sequential server-side aggregation; which term dominates depends on the setting: the sequential server-side decryption of N_{pair} aggregates dominates in the larger-M settings (all FM and CI settings and MN2, e.g., about 91% of the FM1 total), whereas in the small-M setting MN1 (M=3), where each client holds far more samples, the per-client encryption of $\binom{|\mathcal{N}_m|}{2}$ entries dominates (about 70% of the MN1 total). The Stage-1 volume grows quadratically in per-client sample counts, and each 4-byte plaintext expands to a 516-byte ciphertext (129×); BatchCrypt-style aligned packing (Zhang et al., 2020a) (pack64), which places 64 32-bit slots in the 2048-bit plaintext space while preserving slot-wise homomorphic addition, divides the encryption, addition, and decryption counts and the traffic by 64. Stage 1 is executed once, before training: under packing, the estimated THE time ranges from 2.0h (FM1) to 25.8h (MN1), versus 129–1653h unpacked; the plaintext distance computation for the same stage takes 8–186s (Table 21), so the cryptographic layer accounts for essentially all of this one-time cost. This quadratic growth in sample counts is the Stage-1 bottleneck noted in the limitations. Because Stage 1 aggregates every observable pair, Stages 3 and 4 only retrieve already-decrypted entries from their holders; this plaintext retrieval costs 4 B per entry and is omitted from the totals.

Stage-wise wall-clock times. Table 21 decomposes the pipeline into local training (reported per client; local stages run in parallel across clients), the plaintext Stage-1 distance computation, clustering (the MN2 outlier of 5,417.8s stems from the server-side global medoid clustering: the M=10 split pools far more local medoids than MN1's M=3, while per-client local clustering takes under 25s in total), mixup, and server-side training.

Cryptographic unit benchmarks. Table 22 lists the measured Paillier unit operations underlying the THE estimates. The threshold scheme's secret-share partial decryption is not implemented in the library, so standard decryption is used as its proxy (threshold partial decryption performs a modular

Table 20: Stage-1 communication and THE workload per setting (single run, seed fixed, P=1.0). $\Sigma|V|$ = encrypted items, N_{pair} = decrypted aggregated entries; traffic $= 2\,\Sigma|V| \times 516\,\text{B}$ (each encrypted entry is uploaded once and its aggregate is returned to each of its holders); mixup is transmitted as plaintext floats and is unaffected by packing. THE times combine the Paillier unit benchmarks of Table 22 with the analytical operation counts, assuming per-client parallel encryption and sequential server-side addition and decryption. pack64 = aligned packing (Zhang et al., 2020a) of 64 32-bit slots per 2048-bit Paillier plaintext, with values quantized to at most 28 bits so that slot sums over $|V|_{\max} \leq 13$ entries stay within the 32-bit slots.

	$\Sigma\|V\|$	N_{pair}	**Stage-1 traffic (GiB)**		**Mixup (GiB)**	**THE time (h)**	
			naive	pack64		naive	pack64
FM1	3.24×10^8	2.40×10^8	311.3	4.9	0.46	129.2	2.0
FM2	9.00×10^8	6.25×10^8	864.9	13.5	1.34	338.0	5.3
FM3	9.00×10^8	6.39×10^8	864.9	13.5	2.01	345.1	5.4
FM4	9.00×10^8	6.39×10^8	864.9	13.5	2.18	345.1	5.4
FM5	1.30×10^9	8.48×10^8	1245.4	19.5	3.12	461.5	7.2
CI1	5.00×10^8	3.75×10^8	480.5	7.5	18.4	269.7	4.2
CI2	1.12×10^9	7.00×10^8	1081.2	16.9	37.4	536.6	8.4
CI3	2.00×10^9	1.07×10^9	1922.2	30.0	84.5	871.3	13.6
CI4	6.25×10^8	4.30×10^8	600.7	9.4	14.1	344.7	5.4
CI5	8.33×10^8	6.25×10^8	800.9	12.5	14.8	544.3	8.5
MN1	2.02×10^9	1.01×10^9	1943.7	30.4	0.61	1652.6	25.8
MN2	7.69×10^8	5.92×10^8	739.2	11.6	1.41	424.6	6.6

Table 21: Stage-wise wall-clock times in seconds (single run, P=1.0). Local training is per representative client.

	Local (per client)	**Plain distance**	**Clustering**	**Mixup**	**Server training**
FM1	10.5	7.9	68.2	40.7	478.1
FM2	16.3	27.7	103.2	100.0	1097.2
FM3	52.2	25.7	115.3	126.9	3062.9
FM4	65.8	25.8	117.5	134.0	3305.2
FM5	87.8	29.3	141.4	185.4	4886.0
CI1	103.8	52.5	42.8	18.7	1007.1
CI2	153.5	115.1	66.9	37.3	1887.3
CI3	204.8	186.4	109.7	170.7	3847.3
CI4	128.7	66.8	39.0	15.0	695.6
CI5	170.2	91.3	50.1	32.3	837.2
MN1	5985	17.0	105.0	56.8	234.6
MN2	2025	8.4	5417.8	58.2	1238.4

exponentiation of the same order but is severalfold slower and adds share combination (Fouque et al., 2000; Damgård & Jurik, 2001)), making the THE time estimates a lower bound.

Table 22: Paillier unit-operation benchmarks: 2048-bit key, single thread, mean over 1,000 repetitions (phe/python-paillier with gmpy2 acceleration).

Operation	**Mean time**	**Note**
Encryption	6.147 ms	-
Homomorphic addition	9.68 μs	Addition of two ciphertexts
Decryption	1.766 ms	Proxy for threshold partial decryption
Ciphertext size	516 bytes	512-byte ciphertext + 4-byte exponent