Federated Concept-Based Models: Interpretable models with distributed supervision

Dario Fenoglio[1,2]* **Arianna Casanova**[3]* **Francesco De Santis**[4]
Gabriele Dominici[1] **Johannes Schneider**[3] **Pietro Barbiero**[5]
Giovanni De Felice[1] **Marc Langheinrich**[1] **Martin Gjoreski**[1]

[1]Università della Svizzera italiana, Lugano, Switzerland
[2]Massachusetts Institute of Technology, Cambridge, MA, USA
[3]University of Liechtenstein, Vaduz, Liechtenstein
[4]Politecnico di Torino, Turin, Italy
[5]IBM Research, Zurich, Switzerland

Abstract

Concept-based Models (CMs) enhance interpretability in deep learning by grounding predictions in human-understandable concepts. However, concept annotations are costly and rarely available at scale within a single data source. Federated Learning (FL) could alleviate this limitation by enabling cross-institutional training over concept annotations distributed across multiple data owners. Yet, FL lacks interpretable modeling paradigms. Integrating CMs with FL is non-trivial: although FL supports heterogeneous and non-stationary client participation, it typically assumes a fixed shared architecture, whereas CMs may require architectural adaptation as the available concept set evolves. We propose *Federated Concept-based Models* (F-CMs), a new methodology for deploying CMs in evolving FL settings. F-CMs aggregate concept-level information across institutions and efficiently adapt the model architecture to changes in concept supervision while preserving privacy. Empirically, F-CMs maintain accuracy and intervention effectiveness comparable to training settings with full concept supervision, while outperforming on average non-adaptive federated baselines. Notably, F-CMs enable interpretable inference on concepts unavailable to a given institution, a key novelty over existing approaches.

1 Introduction

In many real-world applications, the deployment of machine learning systems is subject to constraints beyond predictive accuracy: practitioners need models whose predictions are interpretable and auditable, providing tools for error diagnosis, fairness assessment, and compliance with legal standards [1–3]. Among existing approaches, *Concept-based Models* (CMs) offer an established framework in which predictions are formulated in terms of high-level, human-interpretable variables, referred to as *concepts* [4–6]. In practice, however, their applicability is constrained by the need for explicit concept supervision during training [7, 8]. Concept annotations are often expensive to obtain, time-consuming, and rarely available at scale within a single data source.

A natural way to alleviate this limitation is to pool concept knowledge across multiple data owners, such as hospitals, research centers, or devices [9, 10]. *Federated Learning* (FL) provides a principled paradigm for collaborative training without sharing raw data, making it particularly appealing in privacy-sensitive settings where concept annotations themselves may encode sensitive information [11, 12]. However, combining CMs with FL creates a fundamental mismatch: FL is designed for

*Equal contribution. Corresponding authors: dario.fenoglio@usi.ch, arianna.casanova@uni.li

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

dynamic settings in which clients may join or leave and client data distributions may differ, while still relying on a shared model architecture with a fixed parameterization. CMs challenge this assumption, as their architecture is explicitly tied to the available concept set and, often, to the dependencies among concepts. As a result, changes that FL can typically absorb at the level of client participation or data distribution may require architectural adaptation in CMs. This issue becomes especially pronounced in realistic FL deployments, where several sources of variation may exist [13–16].

To bridge this gap, we introduce *Federated Concept-based Models (F-CMs)*, a new methodology for deploying concept-based architectures in realistic federated settings. Crucially, F-CMs handle *statistical heterogeneity* and *temporal non-stationarity*: as new clients join (potentially from new data distributions), they may introduce previously unseen concepts and novel dependencies between them. To accommodate this, F-CMs expand the shared concept space and, when available, update the concept dependency structure. Furthermore, the architecture is adapted by updating only the affected components, thereby avoiding full retraining while preserving previously learned knowledge. To the best of our knowledge, F-CMs are the first to address evolving concept spaces under temporal non-stationarity (dynamic participation and data drift) in FL by adapting the model structure, instead of assuming a fixed shared model. Our key contributions benefit both communities and are as follows:

- We introduce **F-CMs**, a novel methodology for deploying **concept-based architectures in realistic federated settings**, addressing heterogeneous, partial concept supervision and temporal non-stationarity.
  - For evolving FL settings, it provides a systematic way to incorporate architectural **interpretability** while allowing the shared model structure to adapt during training, moving beyond the fixed-architecture assumption of standard FL.
  - For interpretability, it **reduces the cost** for concept annotations at a single institution by enabling privacy-preserving knowledge sharing.
- We instantiate F-CMs with four CM architectures and validate them on synthetic benchmarks and two real-world medical imaging datasets. Results show **comparable predictive and intervention accuracy** w.r.t. full concept supervision, enable **interpretable inference over concepts missing at individual clients**, and **reduce retraining cost** under federation growth.

With this work, we hope to contribute toward the broader goal of scaling interpretable models while remaining grounded in real-world constraints and deployment scenarios.

2 Preliminaries

2.1 Concept-based Models (CMs)

CMs predict a task through human-interpretable concepts. In what follows, we consider two main families of CMs, which we refer to as *bipartite CMs* and *graph-based CMs*.

Bipartite CMs, exemplified by [5, 6], map an input $x \in \mathcal{X}$ to a latent representation $z = g(x) \in \mathcal{Z}$, where z is a learned, generally non-interpretable representation of the input. This representation is used to predict a set of m concepts as $\hat{c}_j = h_j(z)$, where each $\hat{c}_j$ may be scalar- or vector-valued. The full concept vector is then given by $\hat{\mathbf{c}} = (\hat{c}_1, \ldots, \hat{c}_m) \in \mathcal{C}$. The task prediction is computed from the concepts and, possibly, the latent input representation $\hat{y} = f(\hat{\mathbf{c}}, z) \in \mathcal{Y}$. This model induces a *directed acyclic graph* (DAG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ [17] over interpretable variables $\mathcal{V} = \{C_1, \ldots, C_m, Y\}$, where edges $\mathcal{E}$ are directed only from concepts to the task, i.e., $C_j \to Y$ for all j.

Graph-based CMs, exemplified by [18, 19], instead allow for dependencies among concepts. In this case, concept and task follow a more general DAG $\mathcal{G}$, where each concept may depend on a set of parent concepts. Concretely, concept and task predictions can be written as $\hat{c}_j = h_j(\hat{\mathbf{c}}_{\mathrm{PA}(C_j)}, z)$ and $\hat{y} = f(\hat{\mathbf{c}}_{\mathrm{PA}(Y)}, z)$, where PA denotes the parent set induced by $\mathcal{G}$. Bipartite CMs can be recovered as a special case of graph-based CMs with only concept-to-task edges.

2.2 Federated Learning (FL)

FL trains a shared model across multiple data owners (*clients*) without sharing raw data [11]. Let $m_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$ denote the shared model with parameters $\mathbf{w}$. The client population may vary over time, denoted by $\mathcal{K}(t)$; at round t, the server selects a subset $\mathcal{K}_t \subseteq \mathcal{K}(t)$ for participation [14, 20]. Each client

$k \in \mathcal{K}(t)$ holds a private dataset $\mathcal{D}^{(k)} = \{(x_i^{(k)}, y_i^{(k)})\}_{i=1}^{n^{(k)}}$. In standard FL, the goal is to minimize a weighted global objective $F(\mathbf{w}) = \sum_{k=1}^{K} p^{(k)} F^{(k)}(\mathbf{w})$, where $F^{(k)}(\mathbf{w}) = \mathbb{E}_{(x,y)\sim\mathcal{D}^{(k)}}[\ell(m_{\mathbf{w}}(x), y)]$ is the local objective of client k and $p^{(k)}$ typically weighs clients proportionally to their data size. At round t, the server broadcasts the current weights $\mathbf{w}_t$ to clients in $\mathcal{K}_t$, each selected client performs local optimization on $\mathcal{D}^{(k)}$, and the server aggregates the returned updates to obtain $\mathbf{w}_{t+1}^{(k)}$, typically via *FedAvg* [11], i.e., by weighted averaging proportional to local dataset sizes.

Statistical heterogeneity and temporal non-stationarity. Practical FL deployments are rarely stationary [21]. First, FL is typically *statistically heterogeneous*: different clients observe different data distributions, i.e., $\exists\, k \neq k'$ s.t. $\mathbb{P}^{(k)}(X, Y) \neq \mathbb{P}^{(k')}(X, Y)$, so their local objectives $F^{(k)}$ differ [15, 22]. Second, FL can be *temporally non-stationary*, meaning that the data and/or the set of participating clients changes over rounds. This commonly arises through (i) *dynamic client participation*, where the available client pool $\mathcal{K}(t)$ and sampled participants $\mathcal{K}_t$ vary over time as clients join or leave [20]; and (ii) *data drift*, where even for a fixed client k, its distribution changes over time, i.e., $\mathbb{P}_t^{(k)}(X, Y) \neq \mathbb{P}_{t'}^{(k)}(X, Y)$ for $t \neq t'$ [16, 23]. Both effects induce a time-varying optimization landscape and can destabilize global updates.

These conditions violate key assumptions of standard CMs. Usually, standard CMs assume a concept set that determines the architecture. In FL settings, however, concept annotations would be distributed across clients and may evolve with clients or data, so concepts and their dependencies can change over time. Consequently, a federated CM must (i) reconcile heterogeneous, partial concept supervision across clients, and (ii) dynamically update concepts, their dependencies and corresponding architecture during training, rather than relying on a fixed design upfront.

2.3 Problem Setting

We study FL of CMs under statistical heterogeneity and temporal non-stationarity. Training proceeds over rounds $t = 1, \ldots, T$, where a subset of clients $\mathcal{K}_t$ participates. Client k holds a local dataset $\mathcal{D}^{(k)} = \{(x_i^{(k)}, \mathbf{v}_i^{(k)})\}_{i=1}^{n^{(k)}}$, where $x_i^{(k)} \in \mathcal{X}$ is the input and $\mathbf{v}_i^{(k)}$ contains the available supervision for sample i, i.e., concept annotations and/or task label. We denote by $\mathcal{V}^{(k)} \neq \emptyset$ the whole set of interpretable variables (concepts and task) supervised by client k. Each client k additionally provides a DAG $\mathcal{G}^{(k)}$, represented by a *weighted adjacency matrix* $A^{(k)} \in [0, 1]^{|\mathcal{V}^{(k)}| \times |\mathcal{V}^{(k)}|}$[24]. Notably, for bipartite CMs, the structure is fixed a priori by allowing only concepts-to-task edges.

Clients may dynamically join or leave the federation over time; when a client leaves, its raw data is discarded, while its contribution to the learned model is retained. Consequently, the global set of supervised variables evolves as $\mathcal{V}_t = \bigcup_{\tau=1}^{t} \bigcup_{k\in\mathcal{K}_\tau} \mathcal{V}^{(k)}$.

The problem is to learn, at each round t, a shared CM $F_{\mathbf{w}_t}$ that minimizes the federated loss induced by the supervision available at each client, while adapting architecture as the global variable set $\mathcal{V}_t$ and its dependencies evolve. Each client contributes only losses for the variables it supervises, so the server must reconcile partial objectives without accessing raw data.

3 Federated Concept-based Models

We now formalize **Federated Concept-based Models (F-CMs)** as a federated training framework for bipartite and graph-based CMs under partial, heterogeneous concept supervision and temporal non-stationarity; Algorithm 1 in App. E.1 summarizes the full procedure. As in standard FL, F-CMs maintain a single shared CM optimized through repeated rounds of client-side training and server-side aggregation. Beyond standard FL, F-CMs support an evolving concept space via three key extensions:

1. **Graph aggregation**: at each round t, the server aggregates client-provided structural information ($A^{(k)}$) into a shared DAG $\mathcal{G}_t$ encoding dependencies and defining the model architecture;
2. **Dynamic architecture adaptation**: the shared CM is decomposed into *concept-specific* and *task-specific* modules, where each module denotes the component used to predict a given concept or the task label. This enables the interpretable architecture to expand and rewire modularly as new concepts or dependencies emerge. For instance, when a client introduces a new concept or a new concept dependency, F-CMs add or modify only the modules associated with the involved concepts, while leaving the remaining components unchanged.

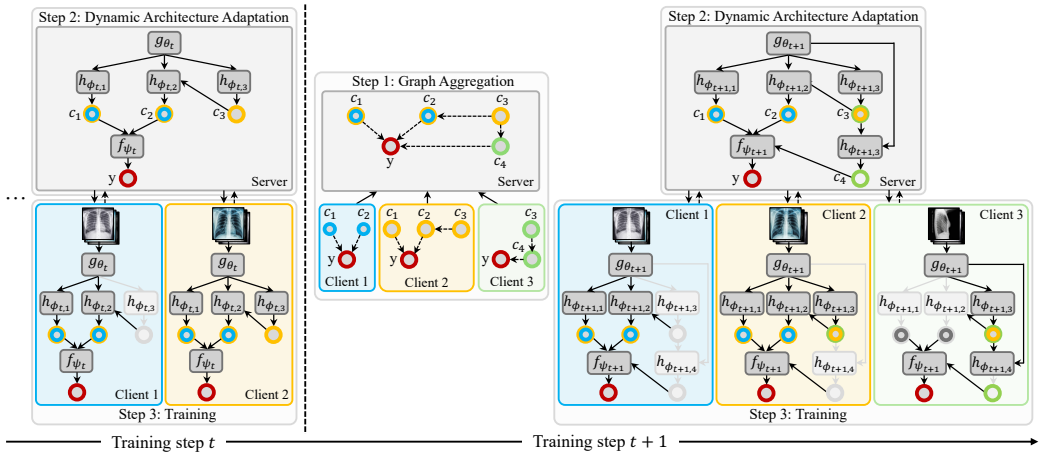


Figure 1: **F-CMs overview.** At each round t, F-CMs perform three steps: (1) *Graph aggregation:* the server combines client graphs into a shared graph $\mathcal{G}_t$; (2) *Architecture adaptation:* the shared CM adapts to $\mathcal{G}_t$ and/or newly observed concepts; (3) *Training:* clients update only supervised concept/task modules (dark grey), leaving others frozen (light grey), and the server aggregates the corresponding updates. The figure shows two rounds, where new clients introduce new concepts.

3. **Module-specific optimization**: each client updates only the modules associated with the interpretable variables it supervises, while modules for unobserved concepts or task labels remain frozen. Server-side aggregation is then performed separately for each module, using only the updates from clients that provided supervision for the corresponding variable.

3.1 F-CM Shared Model

At round t, the server maintains a shared CM (bipartite or graph-based), with parameters $\mathbf{w}_t = (\theta_t, \phi_t, \psi_t)$, which maps inputs x to predictions over variables $\mathcal{V}_t$ through the following modules:

- a *latent encoder* $g_{\theta_t} : \mathcal{X} \to \mathbb{R}^d$ producing $z = g_{\theta_t}(x)$;
- a collection of *concept encoders* $\{h_{\phi_t,j}\}_j$ predicting the value of concepts $C_j \in \mathcal{V}_t \setminus \{Y\}$ as $\hat{c}_j = h_{\phi_t,j}\big(\hat{\mathbf{c}}_{\mathrm{PA}_t(C_j)}, z\big)$;
- a *task decoder* f_{ψ_t} producing $\hat{y} = f_{\psi_t}\big(\hat{\mathbf{c}}_{\mathrm{PA}_t(Y)}, z\big)$;

Here, PA_t denotes the parent set in a shared DAG $\mathcal{G}_t$ constructed over $\mathcal{V}_t$ by aggregating client-specific DAGs $\mathcal{G}^{(k)}$, as described in Section 3.2. In bipartite instantiations, $\mathcal{G}_t$ allows only concept-to-task edges, hence $\mathrm{PA}_t(C_j) = \emptyset$ for every concept C_j, and $\mathrm{PA}_t(Y) = \mathcal{V}_t \setminus \{Y\}$.

Instantiations. F-CMs are agnostic to the specific CM architecture used as the shared model. An instantiation specifies the parameterization of modules $\{h_{\phi_t,j}\}_j$ and f_{ψ_t}.

3.2 F-CM Pipeline

We next detail how F-CMs aggregate structure, adapt the shared architecture, and perform module-specific optimization under partial supervision. Figure 1 provides an overview.

3.2.1 Graph Aggregation

A single client's structural knowledge can be unreliable: local graphs may be noisy due to limited, non-IID data and may also be manipulated by compromised clients (i.e., poisoning) [25]. We therefore aggregate client-provided structures into a shared DAG $\mathcal{G}_t$ over the current global variable set $\mathcal{V}_t$, by favoring edges and directions *with the strongest support across observed clients*.

Specifically, each client k sends a representation of its local DAG $\mathcal{G}^{(k)}$ to the server only upon joining or when its local structure changes, encoded as a weighted adjacency matrix $A^{(k)}$ over its supervised variables $\mathcal{V}^{(k)}$. We interpret $A^{(k)}_{ij}$ as the client's confidence in the directed edge $V_i \to V_j$. For each unordered pair $\{V_i, V_j\} \subseteq \mathcal{V}_t$ with $i \neq j$, we define a categorical distribution over the

three possible outcomes $\{V_i \to V_j,\ V_j \to V_i,\ \varnothing\}$, with probabilities $A^{(k)}_{ij}, A^{(k)}_{ji}, A^{(k)}_{\varnothing,ij}$, respectively. Here, $A^{(k)}_{\varnothing,ij} := 1 - A^{(k)}_{ij} - A^{(k)}_{ji}$ denotes the confidence assigned by client k to the absence of an edge between V_i and V_j. The server aggregates client confidences into a *strength* matrix by summing contributions from all clients observed up to round t, denoted by $\mathcal{K}_{\leq t}$, using the latest graph received from each client:

$$[\bar{A}_t]_{ij} := \sum_{k \in \mathcal{K}_{\leq t}:\, V_i, V_j \in \mathcal{V}^{(k)}} \alpha_k \, A^{(k)}_{ij},$$

where α_k are aggregation weights (e.g., proportional to client dataset sizes, or uniform/reliability-aware; see App. D.10), and $[\bar{A}_t]_{ij} = 0$ if the sum is empty. The same rule is applied to $[\bar{A}_t]_{\varnothing,ij}$. The shared graph $\mathcal{G}_t$ is then constructed by applying a maximum-consensus rule to each pair

$$(V_i \to V_j) \in \mathcal{G}_t \iff [\bar{A}_t]_{ij} = \max\big\{[\bar{A}_t]_{ij}, [\bar{A}_t]_{ji}, [\bar{A}_t]_{\varnothing,ij}\big\}.$$

Ties are broken at random. Cycles are resolved iteratively (Sec. C.2). App. D.6 further shows that graph communication can be equipped with an optional edge-level differentially private mechanism, providing edge-level structural privacy with limited impact on graph quality and predictive performance.

3.2.2 Dynamic Architecture Adaptation

The shared DAG $\mathcal{G}_t$ from the previous step defines the CM connectivity by specifying predicted variables and dependencies. Under non-stationarity, these may change as new clients introduce unseen concepts or revised dependencies. Accordingly, F-CMs adapt the shared CM only when module interfaces change, while preserving all unaffected ones. At round t, this amounts to updating the concept encoders $\{h_{\phi_t,j}\}_j$ and f_{ψ_t}. The latent encoder g_{θ_t} remains unchanged.

(E1) Adding a new concept. For any newly observed concept C_j, the server instantiates a new module $h_{\phi_t,j}$. The task decoder is updated if $C_j \in \mathrm{PA}_t(Y)$, while other modules remain unchanged.

(E2) Updating an edge (add/remove/re-orient). In graph-based F-CMs, changing the relation between two variables modifies parent sets and thus module inputs: adding $C_i \to C_j$ appends $\hat{c}_i$ to the inputs of $h_{\phi,j}$; removing $C_i \to C_j$ removes it; re-orienting $C_i \to C_j$ into $C_j \to C_i$ is treated as remove+add, so $h_{\phi,j}$ loses $\hat{c}_i$ while $h_{\phi,i}$ gains $\hat{c}_j$. The task decoder is updated analogously. In bipartite F-CMs, updates are constrained, disallowing concept-to-concept edges and re-orientations.

More specifically, when the parent set of a node changes, only the module predicting that node is modified. Conversely, changing a node's children does not modify that node's own module, but only the modules predicting the affected children. The concrete resizing operations depend on the chosen CM instantiation and are detailed in App. B.

Warm-start initialization. When the module input expands, only the newly introduced parameters are initialized, while existing parameters are preserved. For example, weights connected to new inputs may be zero-initialized to preserve pre-update behavior, or initialized randomly. Modules whose interfaces are unchanged retain their parameters exactly, avoiding full retraining.

3.2.3 Module-Specific Training and Module-Wise Aggregation

Because concept and task supervision are unevenly distributed across clients, vanilla FedAvg [11] would mix updates for modules that some clients cannot supervise. F-CMs therefore perform module-specific training and module-wise aggregation, restricting contributions to supervised components. This induces a masked federated optimization problem, where each client optimizes a client-specific partial objective over a subset of shared parameters.

Module-specific local optimization. At round t, each client $k \in \mathcal{K}_t$ receives the broadcasted shared CM $(\mathbf{w}_t, \mathcal{G}_t)$, where $\mathbf{w}_t$ denotes the pre-training parameters obtained after architecture adaptation, and performs E local optimization steps using only its available supervision. The local objective is

$$F^{(k)}(\mathbf{w}_t; \mathcal{G}_t) = \mathbb{E}_{(x,\mathbf{v}) \sim \mathcal{D}^{(k)}} \left[\gamma \sum_{C_j \in \mathcal{V}^{(k)} \setminus \{Y\}} \ell_c\big(\hat{c}_j, c_j\big) + (1-\gamma)\, \mathbf{1}_{\{Y \in \mathcal{V}^{(k)}\}}\, \ell_t\big(\hat{y}, y\big) \right], \qquad (1)$$

where ℓ_c and ℓ_t are standard concept and task losses (e.g., cross-entropy), and $\gamma \in [0,1]$. Only supervised modules contribute gradients; unsupervised ones are frozen and need not be transmitted.

Table 1: **Task accuracy** (%) and **concept coverage** (%) with respect to the ground-truth concepts causally relevant for the task, i.e., concepts with a directed path to the task in the ground-truth graph or the available proxy. Methods that improve the average performance are shown in bold. Results are reported as mean ± standard error over five runs.

Setting	Method	ASIA		SACHS		ALARM		INSURANCE		HAILFINDER		SIIM-PN.		CHEXPERT	
		T	C. Cov.	T	C. Cov.	T	C. Cov.	T	C. Cov.	T	C. Cov.	T	C. Cov.	T	C. Cov.
Cent. (Upper Bound)	OpaqNN	80.7±1.0	100	77.7±1.0	100	74.0±0.3	100	76.3±1.1	100	73.0±0.8	100	74.1±0.6	100	72.0±0.3	100
	CBM	80.0±1.0	100	77.3±0.9	100	73.9±0.5	100	76.3±1.4	100	74.2±0.7	100	73.9±0.3	100	70.7±0.5	100
	CEM	80.4±1.1	100	77.3±0.8	100	71.6±0.4	100	76.0±1.2	100	73.3±0.6	100	73.1±0.7	100	66.6±0.7	100
	CGM	79.8±1.2	100	76.2±1.2	100	71.5±0.9	100	73.4±1.1	100	72.2±0.8	100	68.8±0.8	100	71.6±0.3	100
	C^2BM	80.3±1.0	100	77.9±1.0	100	72.8±0.5	100	74.8±0.9	100	73.5±0.6	100	72.8±0.1	100	69.4±0.7	100
Loc.	OpaqNN	56.5±6.7	46.7±3.3	50.2±9.4	52.0±4.9	45.0±11.1	67.0±5.1	19.4±6.2	72.6±6.1	48.1±7.9	63.7±3.5	50.9±8.6	59.4±3.1	57.7±4.7	31.4±5.4
	CBM	47.7±3.8	46.7±3.3	50.1±7.0	52.0±4.9	51.7±7.0	67.0±5.1	30.9±7.6	72.6±6.1	46.6±7.6	63.7±3.5	44.2±6.2	59.4±3.1	57.7±4.2	31.4±5.4
	CEM	67.1±7.4	46.7±3.3	52.2±8.8	52.0±4.9	42.1±11.1	67.0±5.1	40.8±5.3	72.6±6.1	44.1±7.1	63.7±3.5	62.4±1.4	59.4±3.1	57.3±4.7	31.4±5.4
	CGM	64.7±5.1	46.7±3.3	37.0±9.4	52.0±4.9	46.1±8.9	50.4±6.5	31.7±7.2	70.5±7.4	44.1±7.3	63.7±3.5	56.6±6.0	59.4±3.1	60.9±4.9	31.4±5.4
	C^2BM	66.7±7.1	46.7±3.3	37.3±13.8	52.0±4.9	45.0±8.9	50.4±6.5	22.1±12.1	70.5±7.4	43.0±5.6	63.7±3.5	53.8±6.2	59.4±3.1	58.6±4.2	31.4±5.4
External Static FL	FCL	63.9±6.4	N/A	59.5±1.3	N/A	50.1±5.9	N/A	62.3±1.8	N/A	31.7±2.6	N/A	56.5±5.9	N/A	50.9±0.9	N/A
	FedCBM	68.0±4.8	48.6±5.7	48.7±9.4	54.0±5.1	54.8±8.8	70.0±4.1	59.7±8.7	76.9±2.4	51.1±7.4	84.4±2.9	54.6±4.6	63.6±3.7	66.1±4.0	34.3±3.5
S-F-CMs	OpaqNN	74.0±4.9	46.7±3.3	55.2±9.1	56.0±4.0	61.7±7.4	71.3±5.4	64.3±8.5	83.2±2.6	54.0±7.6	84.7±2.8	63.1±1.1	68.8±3.6	56.1±9.0	34.3±3.5
	CBM	63.5±3.6	46.7±3.3	42.2±12.1	56.0±4.0	57.0±8.0	71.3±5.4	58.4±11.8	83.2±2.6	51.8±10.8	84.7±2.8	55.8±6.2	68.8±3.6	67.0±4.2	34.3±3.5
	CEM	72.8±8.0	46.7±3.3	39.9±15.2	56.0±4.0	56.2±10.9	71.3±5.4	67.0±3.4	83.2±2.6	55.1±9.7	84.7±2.8	56.8±6.5	68.8±3.6	54.9±9.2	34.3±3.5
	CGM	70.8±8.3	46.7±3.3	39.3±14.9	56.0±4.0	54.6±10.5	51.3±6.7	58.2±11.8	76.8±5.9	50.6±7.7	83.3±2.4	61.8±0.3	68.8±3.6	65.7±3.8	34.3±3.5
	C^2BM	70.4±7.7	46.7±3.3	46.5±15.5	56.0±4.0	57.1±10.4	51.3±6.7	60.1±13.9	78.9±7.1	59.1±8.2	83.3±2.4	50.0±8.0	68.8±3.6	61.6±4.9	34.3±3.5
F-CMs	OpaqNN	**80.5±1.1**	**100**	**75.2±1.0**	**100**	**72.7±0.7**	**100**	**71.3±0.6**	**100**	**68.0±1.4**	**100**	**64.8±1.0**	**100**	**70.0 ±0.2**	**100**
	CBM	**80.1±0.9**	**100**	**75.2±0.9**	**100**	**71.0±1.9**	**100**	**69.5±0.6**	**100**	**70.6±1.5**	**100**	**67.6±1.1**	**100**	**69.2 ±0.7**	**100**
	CEM	**80.7±1.0**	**100**	**75.7±1.0**	**100**	**72.8±0.7**	**100**	**71.0±0.5**	**100**	**71.6±1.0**	**100**	**68.0±0.1**	**100**	**67.0 ±0.1**	**100**
	CGM	**77.8±1.3**	**100**	**71.7±1.8**	**100**	**68.5±3.3**	**100**	**70.0±1.1**	**100**	**63.8±3.4**	**100**	**63.6±0.2**	**100**	**67.5 ±0.3**	**100**
	C^2BM	**80.4±1.0**	**100**	**76.2±1.0**	**100**	**73.3±0.5**	**100**	**74.2±1.1**	**100**	**72.0±1.8**	**100**	**65.6±2.0**	**100**	**66.1 ±1.3**	**100**

Module-wise aggregation. The server aggregates each module using only the clients that produced updates for that module. We use a FedAvg-like rule, although F-CMs can in principle support other aggregation rules and architectures [26–29]. Let $\mathcal{K}_t^j \subseteq \mathcal{K}_t$ denote the clients updating concept module $h_{\phi_t,j}$, yielding local parameters $\phi_{t+1,j}^{(k)}$. The aggregated parameters are

$$\phi_{t+1,j} = \sum_{k \in \mathcal{K}_t^j} \beta_j^{(k)} \, \phi_{t+1,j}^{(k)}, \qquad \beta_j^{(k)} = \frac{n^{(k)}}{\sum_{r \in \mathcal{K}_t^j} n^{(r)}}.$$

The aggregation weights are normalized independently for each module, so the number of clients supervising one concept does not affect the aggregation weights of other concept modules.

The task module is aggregated analogously over clients that updated it, while modules not updated by any client remain unchanged. Since all participating clients update the latent encoder, its parameters are aggregated with the same FedAvg-like rule over $\mathcal{K}_t$.

Importantly, while only clients with supervision for a concept contribute to updating its prediction module, its predictions can still inform the task decoder and, in graph-based models, downstream concept modules, even at clients without local annotations for that concept.

Theoretical analysis. F-CM training is non-standard: clients may have partial supervision, aggregation is module-wise, and the architecture may change. Appendix E.2 provides a theoretical analysis for a static version of F-CM with fixed clients and concept supervision, together with a piecewise-stationary extension covering changes in the client set, supervision patterns, and architecture at regime boundaries. The static analysis allows us to study, in isolation, the effects of partial supervision and module-wise aggregation on training. Specifically, we show that if a global objective exists under standard assumptions, and the aggregated module-wise updates are sufficiently aligned with its gradient, then the usual nonconvex stationarity behavior is recovered up to a bounded mismatch. Exact alignment gives the standard $O(T^{-1/2})$ rate. We further empirically quantify this alignment mismatch along the optimization trajectory. In the piecewise-stationary setting, the same average-stationarity guarantee is preserved under eventual stabilization, bounded regime-start objective gaps, and uniformly bounded regime-specific constants.

4 Experimental Evaluation

In our experiments, we instantiate F-CMs with two bipartite CMs, CBM [5] and CEM [6], and two graph-based CMs, CGM [18] and C^2BM [19]. CBM predicts scalar concepts, CEM predicts concept embeddings, while CGM and C^2BM condition concept predictions on parent concepts in a DAG (details in App. B). Specifically, C^2BM assumes a *causal* DAG, where each directed edge represents a direct causal relation between variables. By contrast, CGMs can be instantiated with different types of graphs. To provide a unified treatment across graph-based models, we use causal DAGs also for CGMs. This is an implementation choice: F-CMs can also be used with other DAG structures.

We empirically evaluate F-CMs across interpretability and FL objectives. Specifically, we assess (i) task accuracy and concept coverage (Section 4.1), (ii) responsiveness to ground-truth concept interventions (4.2), and (iii) training efficiency under evolving federations with dynamic architecture updates (4.3). The appendix complements this evaluation with extended intervention and adaptation analyses across architectures and instantiations, and analyses of the extent and computational overhead of modular adaptation (D.1–D.5). We also report privacy–utility trade-offs under sample-level differential privacy for model updates and edge-level differential privacy for client graph communication (D.6), sensitivity analyses with respect to the number of clients, selective parameter freezing, the concept-loss weight, client arrival timing, and concept-space expansion (D.7), robustness to inter-client semantic disagreement, missing and noisy concept annotations, and noisy client graphs (D.8 and D.9), and an ablation of graph aggregation strategies under graph perturbations (D.10). Finally, we study a natural extension of F-CMs to multimodal settings, particularly relevant in real-world FL where clients access heterogeneous modalities while sharing common semantic targets; App. D.11 formalizes this extension and reports an experiment on a realistic multimodal dataset.

Baselines and training regimes. We compare F-CM instantiations to an opaque neural baseline (**OpaqNN**) under four training regimes: (i) **Centralized (Cent.)**, pooling all client data and concept annotations, serving as an upper bound; (ii) **F-CMs** (ours); (iii) **S-F-CMs**, a static federated variant of F-CMs that uses the same optimization and structure aggregation, but fixes the concept space and architecture after the first-round client participation (no adaptation); (iv) **Localized (Loc.)**, training independent client models without sharing. We also include two state-of-the-art federated concept-learning baselines, FedCBM [30] and FCL [31] reported separately as **external static FL baselines** since they assume a static federated setting and rely on method-specific concept-learning pipelines.

Federated protocol under temporal non-stationarity. Federated experiments are conducted under *non-stationarity* induced by dynamic client participation and evolving concept supervision. At each round t, the server samples $|\mathcal{K}_t| = 10$ clients. The client pool $\mathcal{K}(t)$ evolves up to 20 clients during training: after a warm-up phase (at $t = 10$ for ASIA, SACHS, and SIIM, and at $t = 20$ otherwise; App. C.1), new clients join with additional concept supervision and, for graph-based F-CMs, potentially new dependencies. Experiments up to $|\mathcal{K}_t| = 100$ are reported in App. D.7.1; additional ablations on client arrival timing and concept-space expansion are reported in D.7.4–D.7.5.

Datasets and data preparation under heterogeneous partial concept supervision. We consider five Bayesian network benchmarks from `bnlearn` [32] (ASIA[33], SACHS [34], INSURANCE [35], ALARM [36], and HAILFINDER [37]), each with a ground-truth DAG, and two real-world chest X-ray datasets: SIIM-PNEUMOTHORAX [38], with concept annotations generated using a medical CLIP model following established practices [39, 19], and CHEXPERT [40], whose labels are extracted from radiology reports. For each dataset, one variable is treated as the task and the remaining as concepts. To model heterogeneous settings, clients are assigned different data distributions and observe partial subsets of concepts. In the centralized setting, graph-based CMs use the ground-truth DAG or a proxy, whereas in federated and localized ones clients access only their local subgraphs and concepts. To model imperfect knowledge (or data poisoning), client graphs are perturbed by randomly adding, removing, or reversing edges (see Apps. A and C.1). When computing accuracy, variables that a model cannot predict under a given training regime (e.g., localized) are assigned chance-level accuracy, defined as the expected accuracy of a uniform random predictor over the corresponding space.

4.1 Task and Concept Coverage

We evaluate *task accuracy* and *concept coverage*, i.e., the fraction of task-relevant concepts predicted by the model. *Concept accuracy* is reported in App. D.1.

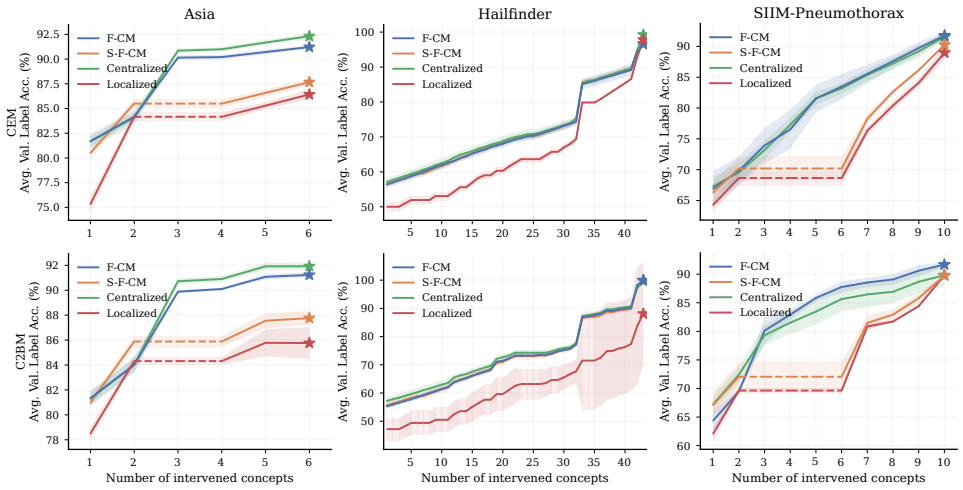


Figure 2: **Label accuracy** (%) on downstream variables (including the task) following interventions on concepts at deeper levels of the ground-truth graph hierarchy. Dashed lines denote interventions that are **not** possible for a given modality due to the absence of predictions for the concept.

Federated aggregation is necessary under statistical heterogeneity. Comparing S-F-CMs to Localized isolates the effect of cross-client aggregation when the concept space is fixed. Across benchmarks, S-F-CMs improve task accuracy and coverage over localized training, showing that aggregating supervision and, for graph-based models, structure is essential to reconcile client heterogeneity.

Aggregation alone is insufficient under temporal non-stationarity; F-CMs close the gap. Under temporal non-stationarity, late-arriving clients introduce additional supervision and, for graph-based models, new dependencies, making a fixed architecture increasingly mismatched to the evolving concept space. This is reflected by the gap between S-F-CMs and both Centralized and F-CMs, and is consistent with the weaker task accuracy of the external static FL baselines FedCBM and FCL. By extending the global concept set and updating the dependency graph, F-CMs adapt the shared architecture and remain close to the centralized bound in task accuracy, while attaining near-100% coverage of task-relevant concepts.

Overall, these results show that realistic federations require (i) graph aggregation to handle heterogeneity and (ii) architecture adaptation to sustain performances as the federation evolves.

4.2 Concept-Level Interventions

In this section, we evaluate responsiveness to *ground-truth concept interventions* by replacing an increasing number of predicted concepts with their ground-truth values, mimicking human corrections at deployment. To ensure a consistent policy, interventions are sampled within progressively deeper levels in the ground-truth DAG hierarchy. Fig. 2 reports the average accuracy over the remaining predicted variables (concepts and task) after each intervention, which we refer to as *label accuracy*. For clearer visualization we fix the client subgraphs across three seeds, isolating intervention effects from variability in the federated partition. Additional results are reported in App. D.

F-CMs enable interventions on locally unannotated concepts. Dashed segments mark interventions that are impossible because the corresponding concepts are not predicted by a given regime. This is the case for both Localized and S-F-CMs: localized models cannot predict concepts outside a client's subgraph, while static federation cannot incorporate concepts introduced by late clients because it fixes the concept space early. By combining cross-client aggregation with dynamic expansion and rewiring, F-CMs predict the broadest set of concepts and keep benefiting from further interventions, yielding higher label accuracy. The same trend holds for task accuracy alone (App. D.2.1).[2]

[2]The Localized regime shows higher variance because, with limited concept supervision, the model may encode information about unobserved concepts into the predicted ones; interventions overwrite these predictions, leading to less stable gains.

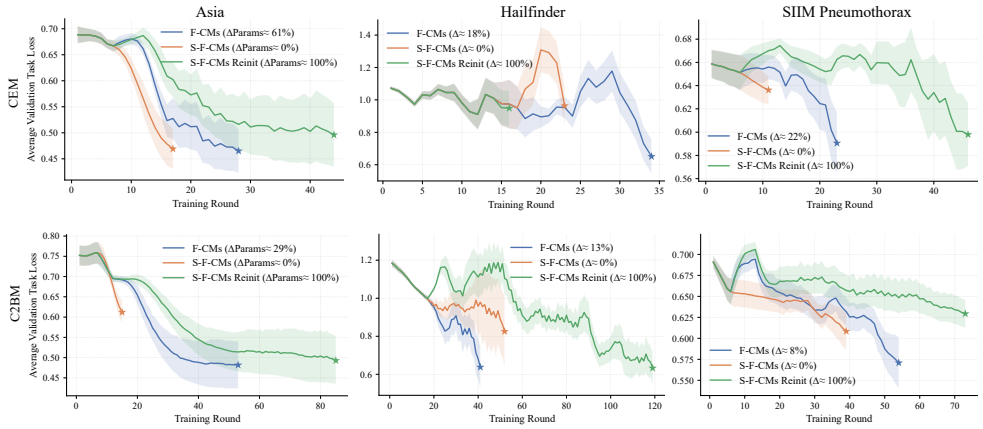


Figure 3: **Convergence under temporal non-stationarity.** Average task loss across rounds for C^{2}BM and CEM, where new clients introduce unseen concepts. F-CMs (ours) adapt the architecture and converge faster and to lower loss than S-F-CMs (no adaptation) and S-F-CMs Reinit (retraining). Curves are truncated at the best loss-round. The legend reports the fraction of modified parameters.

F-CMs achieve interventional responsiveness comparable to centralized. Across datasets and architectures, F-CMs achieve accuracy comparable to centralized setting throughout the intervention trajectory, showing that F-CMs preserve gains from concept corrections as pooled training does.

4.3 Efficient Adaptation in Non-Stationary Federations

We evaluate how F-CMs behave in *non-stationary* federations where new clients join or leave during training and introduce previously unseen concepts, requiring the shared model to expand. Fig.3 reports the average validation task loss over communication rounds on the same datasets and instantiations of the previous experiment. We compare: **F-CMs** (ours), **S-F-CMs** (without architecture adaptation), and **S-F-CMs Reinit** (re-training from scratch whenever new clients join).

Across all datasets and both architectures, F-CMs converge faster and to lower task loss. F-CMs outperform S-F-CMs and avoid the slow recovery of the reinitialization baseline, which erases knowledge accumulated from earlier clients. This translates into greater *adaptation efficiency* in evolving federations: F-CMs modify only the modules affected by newly introduced concepts or dependencies, yielding sparse parameter changes (e.g., 8%–61% across settings), compared to 0% for S-F-CMs (which cannot incorporate new concepts) and 100% for S-F-CMs Reinit. Full results on parameter changes and additional plots for the remaining instantiations are reported in App. D.4 and D.3. These results highlight that modular architecture adaptation yields a more convergence-efficient strategy for continual federation growth, preserving previously learned knowledge while integrating new supervision.

5 Related works

We build on the literature on concept-based models (CMs), which leverage human-interpretable variables for prediction and interpretation. In our experiments, we consider representative architectures from this line of work, including CBM [5], CEM [6], CGM [18], and C^{2}BM [19]. Several other CM variants have also been proposed, e.g., [8, 41, 42]. Our framework is not tied to the architectures evaluated here and could be extended to accommodate other CM variants.

A related line of work addresses the challenge of obtaining concept annotations. Specifically, several CM extensions address the challenge of obtaining concept annotations by relaxing the need for fully observed concept supervision, e.g., by handling missing or noisy labels and combining supervised with unsupervised concept representations [39, 43–46]. However, they typically rely on domain-specific pre-trained models to infer concepts. In contrast, our approach leverages ground-truth concept annotations, even when distributed across multiple entities.

Interpretability in FL remains relatively underexplored. Recent works [30, 31, 47] take initial steps toward combining concepts with FL, but rely on unsupervised concept extraction [48, 49] rather than concept annotations. Consequently, they cannot fully exploit FL to aggregate existing concept-level supervision across clients. They also assume a *static* federation, and therefore do not address *temporal non-stationarity*, common in real-world FL, where new clients may introduce previously unseen concepts and dependencies, requiring the shared concept space and architecture to expand over time.

A related direction is *federated causal discovery*, which studies how to infer causal relationships from decentralized data [50–52]. These methods typically assume a fixed, or pre-specified union, variable set across clients, which is restrictive in settings with evolving concept sets. Moreover, our goal is a model-agnostic graph aggregation mechanism rather than a method tied specifically to causal discovery. Beyond this, *graph federated learning* [53–56] focuses on training graph neural networks via federated optimization, rather than learning evolving interpretable concept-based architectures.

6 Conclusions

We introduced F-CMs, a novel methodology for training concept-based models in realistic federated settings under heterogeneous, partial concept supervision and temporal non-stationarity (dynamic participation and data drift). F-CMs aggregate concept supervision and, when available, concept dependencies across clients throughout training, while adapting the shared model modularly as the concept space evolves. Experiments show that F-CMs preserve task performance while improving concept coverage and intervention effectiveness over local and fixed federated architecture baselines. Importantly, F-CMs enable interpretable inference even for concepts that are locally unannotated.

Limitations and future work. While F-CMs can reuse existing CM architectures, these models were not designed for evolving concept spaces. A promising direction is to develop federation-aware CMs with greater modularity, so that concept and dependency updates affect fewer parameters and enable more efficient adaptation under temporal non-stationarity. Long-running federations could further benefit from principled support-aware retention mechanisms, including concept down-weighting, temporal decay or pruning to limit storage and computation when accumulated concepts remain unsupported for extended periods or become irrelevant, while preserving still useful concepts.

Another important direction is to strengthen the privacy analysis by systematically characterizing information leakage. Although bringing interpretable models into FL is important, it may also introduce new attack surfaces that remain largely unexplored, motivating defenses tailored to concept-based federated models. Beyond this, the modular structure of F-CMs could be exploited to assess client agreement at the concept-module level and develop reliability-aware or robust aggregation strategies against semantic inconsistencies and potentially malicious updates.

Acknowledgments and Disclosure of Funding

We thank Edoardo Gabrielli (Sapienza University of Rome) for carefully reviewing the theoretical analysis in Appendix E.2.

This research was funded by the Swiss National Science Foundation (SNSF), and the European Union's Horizon Europe programme through the projects SmartCHANGE (No. 101080965), and XAI-PAC (No. Z00P2_216405). AC and JS acknowledge support from FFF of the University of Liechtenstein grant lbs_24_08. PB acknowledges support from the SNSF Postdoctoral Fellowships IMAGINE (No. 224226). GDF acknowledges support by the SNSF through the grant 205121_197242 for the project PROSELF and the Hasler Foundation under the Project ID: 2024-05-15-70. The authors declare no competing interests.

References

[1] Margot E Kaminski. The right to explanation, explained. In *Research handbook on information law and governance*, pages 278–299. Edward Elgar Publishing, 2021.

[2] Xuhong Li, Haoyi Xiong, Xingjian Li, Xuanyu Wu, Xiao Zhang, Ji Liu, Jiang Bian, and Dejing Dou. Interpretable deep learning: Interpretation, interpretability, trustworthiness, and beyond. *Knowledge and Information Systems*, 64(12):3197–3234, 2022.

[3] Tribikram Dhar, Nilanjan Dey, Surekha Borra, and R Simon Sherratt. Challenges of deep learning in medical image analysis—improving explainability and trust. *IEEE Transactions on Technology and Society*, 4(1):68–75, 2023.

[4] Bernhard Ganter and Rudolf Wille. *Formal concept analysis*, volume 150. Springer, 1999.

[5] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR, 2020.

[6] Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, et al. Concept embedding models: Beyond the accuracy-explainability trade-off. *Advances in neural information processing systems*, 35:21400–21413, 2022.

[7] Nicola Debole, Pietro Barbiero, Francesco Giannini, Andrea Passerini, Stefano Teso, and Emanuele Marconato. If concept bottlenecks are the question, are foundation models the answer? *arXiv preprint arXiv:2504.19774*, 2025.

[8] Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. Concept-based explainable artificial intelligence: A survey. *ACM Computing Surveys*, 2023.

[9] Geert Litjens, Peter Bandi, Babak Ehteshami Bejnordi, Oscar Geessink, Maschenka Balkenhol, Peter Bult, Altuna Halilovic, Meyke Hermsen, Rob van de Loo, Rob Vogels, Quirine F Manson, Nikolas Stathonikos, Alexi Baidoshvili, Paul van Diest, Carla Wauters, Marcory van Dijk, and Jeroen van der Laak. 1399 H&E-stained sentinel lymph node sections of breast cancer patients: The CAMELYON dataset. *GigaScience*, 7(6):giy065, June 2018. ISSN 2047-217X. doi: 10.1093/gigascience/giy065.

[10] Georgios A. Kaissis, Marcus R. Makowski, Daniel Rückert, and Rickmer F. Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, June 2020. ISSN 2522-5839. doi: 10.1038/s42256-020-0186-1.

[11] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data, January 2023.

[12] Dario Fenoglio, Martin Gjoreski, Edoardo Gabrielli, and Marc Langheinrich. Security and Privacy in Federated Learning: A Survey and Unified Taxonomy of Threat Models, Attacks, and Defenses, May 2026.

[13] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B. Gibbons. Federated Learning under Distributed Concept Drift. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 5834–5853. PMLR, April 2023.

[14] Rafael O. Jarczewski, Eduardo Cerqueira, Luiz F. Bittencourt, Antonio A. F. Loureiro, Leandro A. Villas, and Allan M. de Souza. Let's Federate - Effective Communication Strategy for Dynamic Client Participation. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 361–368, December 2024. doi: 10.1109/ICMLA61862.2024.00055.

[15] Dario Fenoglio, Mohan Li, Pietro Barbiero, Nicholas D. Lane, Marc Langheinrich, and Martin Gjoreski. FLUX: Efficient Descriptor-Driven Clustered Federated Learning under Arbitrary Distribution Shifts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

[16] Mohan Li, Dario Fenoglio, Martin Gjoreski, and Marc Langheinrich. Federated Learning with Profile Mapping under Distribution Shifts and Drifts. In *The Fourteenth International Conference on Learning Representations*, October 2026.

[17] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

[18] Gabriele Dominici, Pietro Barbiero, Mateo Espinosa Zarlenga, Alberto Termine, Martin Gjoreski, Giuseppe Marra, and Marc Langheinrich. Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning. In *The Thirteenth International Conference on Learning Representations*, October 2024.

[19] Giovanni De Felice, Arianna Casanova Flores, Francesco De Santis, Silvia Santini, Johannes Schneider, Pietro Barbiero, and Alberto Termine. Causally reliable concept bottleneck models. *arXiv preprint arXiv:2503.04363*, 2025.

[20] Rafael De Oliveira Jarczewski, Eduardo Cerqueira, Luiz Fernando Bittencourt, Antonio A. F. Loureiro, Leandro A. Villas, and Allan M. De Souza. Participation is Power: Effective Approach to Dynamic Federated Learning. In *Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing*, UCC '25, pages 1–6. Association for Computing Machinery, December 2025. ISBN 979-8-4007-2285-1. doi: 10.1145/3773274.3774701.

[21] Ming Xiang, Stratis Ioannidis, Edmund Yeh, Carlee Joe-Wong, and Lili Su. Efficient Federated Learning against Heterogeneous and Non-stationary Client Unavailability. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

[22] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, et al. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, June 2021. ISSN 1935-8237. doi: 10.1561/2200000083.

[23] Kunjal Panchal, Sunav Choudhary, Subrata Mitra, Koyel Mukherjee, Somdeb Sarkhel, Saayan Mitra, and Hui Guan. Flash: Concept Drift Adaptation in Federated Learning. *Fortieth International Conference on Machine Learning*, June 2023.

[24] Albert-László Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987), 2013.

[25] Gan Sun, Yang Cong, Jiahua Dong, Qiang Wang, Lingjuan Lyu, and Ji Liu. Data Poisoning Attacks on Federated Machine Learning. *IEEE Internet of Things Journal*, 9(13):11365–11375, July 2022. ISSN 2327-4662. doi: 10.1109/JIOT.2021.3128646.

[26] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[27] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=LkFG3lB13U5`.

[28] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[29] Dario Fenoglio, Pasquale Polverino, Akash Dhasade, Jacopo Quizi, Martin Gjoreski, and Marc Langheinrich. ERIS: Enhancing the privacy-utility-scalability trade-off in federated learning via federated shard aggregation. In *Advances in Neural Information Processing Systems*, 2026. URL `https://arxiv.org/abs/2602.08617`.

[30] Jianan Yang and Guodong Long. Concept-Guided Interpretable Federated Learning. In Tongliang Liu, Geoff Webb, Lin Yue, and Dadong Wang, editors, *AI 2023: Advances in Artificial Intelligence*, pages 160–172, Singapore, 2024. Springer Nature. ISBN 978-981-99-8391-9. doi: 10.1007/978-981-99-8391-9_13.

[31] Jiaxin Shen, Xiaoyi Tao, Liangzhi Li, Zhiyang Li, and Bowen Wang. Explaining Federated Learning Through Concepts in Image Classification. In Zahir Tari, Keqiu Li, and Hongyi Wu, editors, *Algorithms and Architectures for Parallel Processing*, pages 325–340, Singapore, 2024. Springer Nature. ISBN 978-981-97-0808-6. doi: 10.1007/978-981-97-0808-6_19.

[32] Marco Scutari. Learning bayesian networks with the bnlearn r package. *Journal of Statistical Software*, 35(i03), 2010.

[33] S. L. Lauritzen and D. J. Spiegelhalter. Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 50(2):157–224, 1988.

[34] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721): 523–529, 2005.

[35] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

[36] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In *Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine*, pages 247–256. Springer-Verlag, 1989. URL `https://www.cse.huji.ac.il/~galel/Repository/Datasets/alarm/alarm.htm`.

[37] Bruce Abramson, John Brown, Ward Edwards, Allan Murphy, and Robert L Winkler. Hailfinder: A bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12(1): 57–71, 1996.

[38] Kihyun You, Jawook Gu, Jiyeon Ham, Beomhee Park, Jiho Kim, Eun K Hong, Woonhyuk Baek, and Byungseok Roh. Cxr-clip: Toward large scale chest x-ray language-image pre-training. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 101–111. Springer, 2023.

[39] Tuomas Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. Label-free Concept Bottleneck Models. In *The Eleventh International Conference on Learning Representations*, September 2022.

[40] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):590–597, July 2019. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.3301590.

[41] Francesco De Santis, Philippe Bich, Gabriele Ciravegna, Pietro Barbiero, Danilo Giordano, and Tania Cerquitelli. Linearly-interpretable concept embedding models for text analysis. *Machine Learning*, 114(10):224, 2025.

[42] Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lió, Frederic Precioso, Mateja Jamnik, and Giuseppe Marra. Interpretable neural-symbolic concept reasoning. In *International Conference on Machine Learning*, pages 1801–1825. PMLR, 2023.

[43] Yoshihide Sawada and Keigo Nakamura. Concept Bottleneck Model With Additional Unsupervised Concepts. *IEEE Access*, 10:41758–41765, 2022. ISSN 2169-3536. doi: 10.1109/ACCESS.2022.3167702.

[44] Emiliano Penaloza, Tianyue H. Zhang, Laurent Charlin, and Mateo Espinosa Zarlenga. Addressing Concept Mislabeling in Concept Bottleneck Models Through Preference Optimization, September 2025.

[45] Lijie Hu, Tianhao Huang, Huanyi Xie, Xilin Gong, Chenyang Ren, Zhengyu Hu, Lu Yu, Ping Ma, and Di Wang. Semi-supervised Concept Bottleneck Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2110–2119, 2025.

[46] Yang Liu, Tianwei Zhang, and Shi Gu. Hybrid Concept Bottleneck Models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 20179–20189, 2025.

[47] Yanci Zhang and Han Yu. LR-XFL: Logical Reasoning-Based Explainable Federated Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19):21788–21796, March 2024. ISSN 2374-3468. doi: 10.1609/aaai.v38i19.30179.

[48] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[49] Bowen Wang, Liangzhi Li, Yuta Nakashima, and Hajime Nagahara. Learning bottleneck concepts in image classification. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 10962–10971, 2023.

[50] Ignavier Ng and Kun Zhang. Towards federated bayesian network structure learning with continuous optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 8095–8111. PMLR, 2022.

[51] Jianli Huang, Xianjie Guo, Kui Yu, Fuyuan Cao, and Jiye Liang. Towards privacy-aware causal structure learning in federated setting. *IEEE Transactions on Big Data*, 9(6):1525–1535, 2023.

[52] Loka Li, Ignavier Ng, Gongxu Luo, Biwei Huang, Guangyi Chen, Tongliang Liu, Bin Gu, and Kun Zhang. Federated causal discovery from heterogeneous data. In *The Twelfth International Conference on Learning Representations*, 2024.

[53] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. *Advances in neural information processing systems*, 34: 6671–6682, 2021.

[54] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S Yu, Yu Rong, et al. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145*, 2021.

[55] Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. Federated graph neural networks: Overview, techniques, and challenges. *IEEE transactions on neural networks and learning systems*, 2024.

[56] Sungwon Kim, Yoonho Lee, Yunhak Oh, Namkyeong Lee, Sukwon Yun, Junseok Lee, Sein Kim, Carl Yang, and Chanyoung Park. Subgraph federated learning for local generalization. *arXiv preprint arXiv:2503.03995*, 2025.

[57] Finn V Jensen and Thomas D Nielsen. *Bayesian Networks and Decision Graphs: February 8, 2007*. Springer, 2007.

[58] Sonia Laguna, Ričards Marcinkevičs, Moritz Vandenhirtz, and Julia E Vogt. Beyond concept bottleneck models: How to make black boxes intervenable? *Advances in neural information processing systems*, 37:85006–85044, 2024.

[59] Stanford AIMI. CheXpert Plus, 2026. URL `https://stanford.redivis.com/datasets/5yyj-1a9f6ap0x?v=1.0`.

[60] Benedikt Boecking, Naoto Usuyama, Shruthi Bannur, Daniel C. Castro, Anton Schwaighofer, Stephanie Hyland, Maria Wetscherek, Tristan Naumann, Aditya Nori, Javier Alvarez-Valle, Hoifung Poon, and Ozan Oktay. Making the most of text semantics to improve biomedical vision-language processing, 2022. URL `https://arxiv.org/abs/2204.09817`.

[61] Bowen Wang, Liangzhi Li, Yuta Nakashima, and Hajime Nagahara. Learning Bottleneck Concepts in Image Classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10962–10971, Vancouver, BC, Canada, June 2023. IEEE. ISBN 979-8-3503-0129-8. doi: 10.1109/CVPR52729.2023.01055.

[62] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[63] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[64] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[65] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32, 2019.

[66] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. Flower: A Friendly Federated Learning Research Framework, March 2022.

[67] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

[68] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[69] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9 (3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[70] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-Friendly Differential Privacy Library in PyTorch, August 2022. Privacy in Machine Learning (PriML) workshop, NeurIPS 2021.

[71] Dario Fenoglio, Gabriele Dominici, Pietro Barbiero, Alberto Tonda, Martin Gjoreski, and Marc Langheinrich. Federated Behavioural Planes: Explaining the Evolution of Client Behaviour in Federated Learning. *Advances in Neural Information Processing Systems*, 37:112777–112813, 2024.

[72] Ulle Endriss and Umberto Grandi. Graph aggregation. *Artificial Intelligence*, 245:86–114, April 2017. ISSN 0004-3702. doi: 10.1016/j.artint.2017.01.001.

[73] José M. Puerta, Juan A. Aledo, José A. Gámez, and Jorge D. Laborda. Efficient and accurate structural fusion of Bayesian networks. *Information Fusion*, 66:155–169, February 2021. ISSN 1566-2535. doi: 10.1016/j.inffus.2020.09.003.

Table of Contents

A Dataset details **17**
- A.1 Bayesian Network datasets (bnlearn) . . . 17
- A.2 SIIM-PNEUMOTHORAX . . . 17
- A.3 CHEXPERT . . . 18
- A.4 CHEXPERT PLUS . . . 18

B Baseline details **19**

C Experimental details **20**
- C.1 Details on data preparation . . . 20
- C.2 Training details . . . 21
- C.3 Licenses and hardware . . . 22

D Additional experiments **23**
- D.1 Concept accuracy . . . 23
- D.2 Concept-level interventions - further experiments . . . 23
- D.3 Additional model instances for efficient adaptation in non-stationary federations . 27
- D.4 Extent of architecture updates under temporal non-stationarity . . . 27
- D.5 Computational overhead of graph aggregation and modular adaptation . . . 28
- D.6 Differential privacy . . . 31
- D.7 Sensitivity analysis . . . 33
- D.8 Robustness to imperfect concept supervision . . . 36
- D.9 Robustness to graph perturbations . . . 38
- D.10 Ablation of graph aggregation strategies under graph perturbations . . . 39
- D.11 Multimodal extension . . . 42

E Algorithmic details and analysis **45**
- E.1 Algorithm . . . 45
- E.2 Theoretical analysis . . . 45

A Dataset details

A.1 Bayesian Network datasets (bnlearn)

We use five benchmark *Bayesian networks* from the `bnlearn` repository [32]. A Bayesian network (BN) [57] is a probabilistic graphical model specified by a DAG and a set of conditional probability distributions (CPDs). The DAG encodes the dependency structure among random variables, while each CPD specifies the distribution of a variable conditional on its parents in the graph. The selected BNs span diverse application domains and vary in graph size and structural complexity.

- ASIA*:* A small network for lung disease diagnosis with variables representing patient symptoms and conditions.
- SACHS*:* A widely used network capturing dependencies among protein and phospholipid expression levels in human cells.
- ALARM*:* A medical network designed to provide an alarm message system for patient monitoring.
- INSURANCE*:* A network for car insurance risk assessment with variables covering driver characteristics, vehicle properties, and accident-related factors.
- HAILFINDER*:* A weather forecasting network with variables for predicting severe weather conditions in northeastern Colorado, including atmospheric measurements and storm indicators.

Table 2: Summary of bnlearn Bayesian networks used in experiments.

Bayesian network	**Nodes**	**Edges**	**Domain**
ASIA	8	8	Medical diagnosis
SACHS	11	17	Protein signaling
ALARM	37	46	Medical monitoring
INSURANCE	27	52	Insurance risk
HAILFINDER	56	66	Weather prediction

For each network, we generate a dataset with a network-specific number of samples, selected via preliminary experiments to ensure stable performance: 15,000 for ASIA, 15,000 for SACHS, 10,000 for ALARM, 20,000 for INSURANCE, and 20,000 for HAILFINDER. Each dataset is split into training, validation, and test sets using a 70%–10%–20% split. For each network, we generate a dataset with a network-specific sample size, selected through preliminary experiments to ensure stable performance: 15,000 for ASIA, 15,000 for SACHS, 10,000 for ALARM, 20,000 for INSURANCE, and 20,000 for HAILFINDER. Each dataset is split into training, validation, and test sets using a 70%–10%–20% split. We interpret the networks causally and treat their DAGs as ground-truth causal graphs. Each directed edge represents a direct causal relation from a parent to a child variable, and the DAG defines the reference causal structure for evaluation. The nodes in each network are treated either as concepts $\{C_j\}_j$ or as the task variable Y. Thus, the generated datasets provide annotations for semantic variables, but no corresponding raw inputs x_i. To construct inputs for concept-based models, concept values are flattened and passed through an autoencoder trained with MSE loss, consisting of two encoder and two decoder layers. The latent dimensionality scales with graph size (ASIA: 32, SACHS: 32, INSURANCE: 64, ALARM: 64, HAILFINDER: 64). The resulting embeddings are perturbed by mixing 50% of the encoded signal with 50% Gaussian noise, and the final inputs are standardized. These transformations yield non-trivial input representations, requiring models to recover the underlying concepts rather than trivially accessing them. The selected tasks are: *dysp* (ASIA), *Akt* (SACHS), *PropCost* (INSURANCE), *BP* (ALARM), and *R5Fcst* (HAILFINDER).

A.2 SIIM-PNEUMOTHORAX

The SIIM-PNEUMOTHORAX dataset is derived from the publicly available chest radiograph collection released by the National Institutes of Health (NIH). It consists of chest X-ray images annotated with binary labels indicating the presence or absence of Pneumothorax. In our experiments, we use the training annotations from the SIIM-ACR Pneumothorax Segmentation challenge on Kaggle and the

corresponding training images from the SIIM PNG image dataset[3]. We then split the resulting dataset into training, validation, and test sets using standard 70%–10%–20% partition.

As the dataset does not provide concept-level annotations, we generate concept labels automatically. Specifically, we use a pretrained CXR-CLIP model [38] following a procedure similar to Oikarinen et al. [39]: candidate concepts are generated once and filtered using the training data, then image-text cosine similarities are computed between each radiograph and the generated concepts to generate annotations. These similarity scores are further binarized independently for each concept using two-cluster k-means, to obtain final binary concept annotations used in our experiments.

As encoder, we use a ResNet-18 followed by a trainable MLP head that maps image representations to the latent space used by the concept and task modules.

A.3 CHEXPERT

We use the CHEXPERT dataset of chest X-ray images annotated with expert-labeled radiological observations [40] hosted on Kaggle[4]. In our setup, the 13 observation labels are used as concept annotations, namely *Enlarged Cardiomediastinum*, *Cardiomegaly*, *Lung Opacity*, *Lung Lesion*, *Edema*, *Consolidation*, *Pneumonia*, *Atelectasis*, *Pneumothorax*, *Pleural Effusion*, *Pleural Other*, *Fracture*, and *Support Devices*. The downstream binary task is derived from the *No Finding* label, which we invert so that the target indicates the presence of any abnormal finding.

For preprocessing, we first merge the provided training and validation metadata files, remove duplicate patients by keeping only the last available study for each subject, and replace missing annotations with 0 and uncertain concept annotations (encoded as -1) with 1. We further filter the dataset by acquisition view, keeping only frontal images. We then shuffle the resulting dataset and create custom training, validation, and test splits using a 70%–10%–20% partition. Following an image preprocessing similar to Laguna et al. [58], each radiograph is center-cropped to a square region, resized to 224×224, histogram-equalized, and converted to grayscale for compatibility with the image encoder. During training, we additionally apply random affine transformations and random horizontal flipping; for validation and test images, only resizing and tensor conversion are used.

As encoder, we again use a ResNet-18, as in Laguna et al. [58], followed by a trainable MLP head.

To obtain a dataset size comparable to the other benchmarks, we further subsample an equal number of examples per task class. Importantly, we preserve the federated heterogeneity by assigning different subsets of concept supervision to different clients, as in the other datasets.

A.4 CHEXPERT PLUS

We use CHEXPERT PLUS, a multimodal extension of CHEXPERT that provides radiology reports paired with chest X-ray images [59]. For the images, we rely on the version of the CHEXPERT dataset hosted on Kaggle, which we pair with the additional reports from CHEXPERT PLUS (see Section A.3). To increase the number of samples with both modalities, we do not filter images by view and we retain duplicate studies from the same patients. However, we perform a patient-level split: all images and reports associated with a given patient are assigned to only one of the training, validation, or test sets. We then apply the same image preprocessing as in the original CHEXPERT setup with a usual split of 70%–10%–20% for training, validation and test.

For the text modality, each radiology report is tokenized and the resulting sequence is used as input to the text encoder (see App. D.11).

The resulting dataset preserves the same semantic prediction targets as CHEXPERT, with each instance associated with both an image and a text input.

As encoders, we use a ResNet-18 for images and the pre-trained CXR-BERT specialized model, a chest X-ray domain-specific language model [60], for text. The outputs of both encoders are then passed through trainable MLP heads.

[3] `https://www.kaggle.com/competitions/siim-acr-pneumothorax-segmentation`; `https://www.kaggle.com/datasets/abhishek/siim-png-images`

[4] `https://www.kaggle.com/datasets/ashery/chexpert`

B Baseline details

All the baselines considered share the same input encoder. They differ only in the parameterization of the concept predictors and the task decoder, and in whether they use a bipartite graph (CBM/CEM) or a more general DAG $\mathcal{G}$ connecting concepts and task.

Concept Bottleneck Models (CBMs) [5]. CBMs use a bipartite graph. Specifically, each concept is predicted independently from an encoder representation z:

$$\hat{c}_j = h_{\phi_j}(z), \text{ for every } j$$

where h_{ϕ_j} is an MLP head outputting either logits/probabilities (categorical concepts) or real values (continuous concepts). The task decoder operates only on the predicted concepts $\hat{\mathbf{c}} = \{\hat{c}_1, ..., \hat{c}_m\}$,

$$\hat{y} = f_\psi(\hat{\mathbf{c}}),$$

implemented as an MLP over concatenation of all concept predictions (no direct dependence on z).

Concept Embedding Models (CEMs) [6]. CEMs share the same bipartite construction as CBMs, but represent each concept with an embedding rather than a scalar. For each concept C_j, the model first produces two concept-specific embeddings from the encoder representation z:

$$\hat{c}_j^+ = \varphi_{\phi_j}^+(z) \in \mathbb{R}^r, \qquad \hat{c}_j^- = \varphi_{\phi_j}^-(z) \in \mathbb{R}^r,$$

where $\hat{c}_j^+$ encodes the *active* state and $\hat{c}_j^-$ the *inactive* state of concept C_j. Then, a scoring function maps their joint space to a concept activity probability,

$$\hat{p}_j \;=\; s_\phi([\hat{c}_j^+, \hat{c}_j^-]^T) \;=\; \sigma\big(W_s[\hat{c}_j^+, \hat{c}_j^-]^T + b_s\big) \in [0, 1],$$

and the final concept embedding fed to the task is the corresponding mixture:

$$\tilde{c}_j \;=\; \hat{p}_j\, \hat{c}_j^+ \;+\; (1 - \hat{p}_j)\, \hat{c}_j^- \;\in \mathbb{R}^r.$$

The task decoder then operates on the concatenation of mixed concept embeddings $\hat{\mathbf{c}}$,

$$\hat{y} \;=\; f_\psi(\hat{\mathbf{c}}),$$

implemented as an (interpretable) linear layer or a small MLP.

For both CBMs and CEMs, changes to the available concept set are handled by adapting the task decoder to the updated input dimensionality and reinitializing the affected parameters. Concept modules associated with existing concepts are reused, while modules for newly added concepts are created and initialized.

Concept Graph Models (CGM) [18]. We implement CGMs using the aggregated DAG produced by our methodology. For each concept node C_j, we predict

$$\hat{c}_j = h_{\phi,j}\big(\hat{\mathbf{c}}_{\mathrm{PA}(C_j)}, z\big),$$

where $h_{\phi,j}$ is an MLP that takes as input the concatenation of the embeddings of the parent concepts of C_j and the encoder representation z, following a formulation similar to CEM. The task prediction is computed analogously,

$$\hat{y} = f_\psi\big(\hat{\mathbf{c}}_{\mathrm{PA}(Y)}, z\big).$$

This factorization ensures that changes to upstream concepts propagate to downstream concepts and the task through the parent-conditioned modules.

Under dynamic architectural adaptation, changes in $\mathrm{PA}(C_j)$ or $\mathrm{PA}(Y)$ are handled by adapting $h_{\phi,j}$ or the task decoder, respectively, resizing the input layer when necessary and reinitializing the affected parameters. Concept/task modules not affected by any change are reused while modules for newly added concepts are created and initialized.

Causally Reliable Concept Bottleneck Models (C^{2}BMs) [19]. C^{2}BMs also use our aggregated DAG, but explicitly separate *exogenous* information extracted from the input from the *structural* mechanism mapping parent variables to each node.

Concretely, each node V in the DAG is equipped with a CEM-style module that outputs an embedding u_V representing the node-specific exogenous signal extracted from the input. This exogenous

representation is used (a) to predict *root* nodes directly via a lightweight output head, and (b) to *parameterize* the parent-to-node mechanism for *non-root* nodes through a hypernetwork. In particular, for non-root nodes we first generate node-specific coefficients from u_V,

$$\theta_V = r_{\phi_V}(u_V),$$

and then predict V from its parents with a linear-in-parents map followed by the appropriate output nonlinearity:

$$\hat{v} = \sigma_V\Big(\langle \theta_V,\ \hat{\mathbf{v}}_{\mathrm{PA}(V)} \rangle\Big) \ \equiv\ f_{\theta_V}\big(\hat{\mathbf{v}}_{\mathrm{PA}(V)}\big),$$

where r_{ϕ_V} is an MLP (hypernetwork) and σ_V is chosen according to the variable type (e.g., sigmoid/softmax/identity). The task node Y is treated identically as a node in the graph.

In C^{2}BM, a change in $\mathrm{PA}(V)$ is handled by resizing the output layer of the corresponding hypernetwork r_{ϕ_V} to generate the updated set of parent-specific coefficients and reinitializing its parameters accordingly. Concept encoders for newly added concepts are also created and initialized.

Federated Concept Learning (FCL) [31]. FCL is a *static* federated concept-learning method built on top of a Bottleneck Concept Learner (BotCL) [61]. On each client, BotCL extracts a set of unsupervised concept activations from the input and uses them for classification through a concept–class score matrix. In the federated setting, clients do not exchange concept predictors or structural relations; instead, each client uploads a matrix of *concept–class co-occurrence scores*, encoding how strongly each learned concept contributes to each class. The server aggregates these scores across clients with a FedAvg-style procedure, after first trimming the highest and lowest values for each entry to mitigate the effect of malicious clients, and sends the aggregated score matrix back to the clients for the next round. Classification is then performed only from the learned concept activations and the aggregated concept–class scores. Since FCL assumes a single shared static concept space and communicates only concept–class scores, it is reported in our experiments as an external *static FL* concept-learning baseline.

Federated Concept Bottleneck Models (FedCBM) [30]. FedCBM is a *static* federated concept-learning method based on a concept bottleneck architecture with *pre-defined* human concepts. The method follows a two-stage pipeline. First, clients collaboratively train a shared *concept bank* from local concept datasets, where each concept is represented by a *Concept Activation Vector* (CAV), i.e., a linear hyperplane normal learned with a federated SVM-like optimization procedure. These shared CAVs define a common concept space across clients. Second, for a standard input x, each client extracts a latent representation $h(x)$ with a backbone encoder, projects it onto each CAV to obtain a vector of concept scores, and feeds this concept vector to a local linear classifier for prediction. Thus, FedCBM performs classification exclusively through a bottleneck of concept scores, while concepts are aligned across clients through the shared concept bank. Since FedCBM assumes a fixed shared concept vocabulary and does not model evolving concept dependencies or graph structure, we report it as an external *static FL* concept-learning baseline.

C Experimental details

C.1 Details on data preparation

SIIM-PNEUMOTHORAX ground-truth graph. The ground-truth graph among concepts and the task for SIIM-PNEUMOTHORAX is constructed following a procedure similar to the one provided in De Felice et al. [19]. First, the Greedy Equivalence Search (GES) algorithm [62], a causal discovery method that infers causal structure from observational data, is applied to the concept and task annotations to produce an initial graph. This graph however may still contain unoriented edges. To resolve these ambiguities, we employ a large language model (LLM), specifically `GPT-4o` [63], enhanced with retrieval-augmented generation (RAG) [64], which uses domain-specific medical literature to provide context. This process enriches the initial graph with background knowledge, yielding the final DAG.

Client local data. In our setting, each client k holds a local dataset of inputs and semantic variable annotations,

$$\mathcal{D}^{(k)} \coloneqq \{(x_i^{(k)}, \mathbf{v}_i^{(k)})\}_{i=1}^{n_k}.$$

A client may also provide a local DAG $\mathcal{G}^{(k)}$ encoding relationships among its observed concepts and the task.

To generate client data, we proceed as follows. Given a dataset $\mathcal{D}$ with full inputs, concepts, and task annotations (see App. A), we first construct a ground-truth causal DAG $\mathcal{G}$ over the available concepts and the task variable (see *Proxy DAG construction and cycle resolution* in App.C.2). Next, we generate $\mathcal{K}$ subgraphs $\{\mathcal{G}^{(k)}\}_{k=1}^{\mathcal{K}}$ of $\mathcal{G}$, where $\mathcal{K}$ denotes the total number of clients considered throughout training. Each $\mathcal{G}^{(k)}$ is defined as an induced subgraph over a subset of nodes, corresponding to a subset of concepts (and possibly the task variable). Specifically, each subgraph is built as follows: starting from the task node Y (a leaf in the graph $\mathcal{G}$), we repeatedly move *upstream* by following incoming edges and collect multiple ancestor paths up to the roots; we then take the union of these paths (equivalently, the induced ancestor tree of Y). To increase heterogeneity across clients, we additionally sample a set of random concept nodes that are not contained in the set of roots of $\mathcal{G}$ and, for each of them, collect one or more ancestor paths up to the roots; these paths are finally merged with the subgraph built from Y. The task node Y is then randomly excluded from a subset of clients to model missing task supervision. We ensure full concept coverage at T for F-CMs, i.e., the union of the client subgraphs in the final federation considered in our experiments covers all concepts and the task variable of the original dataset. To simulate imperfect client knowledge of concept relations, we also perturb each $\mathcal{G}^{(k)}$ by modifying a fraction p of its edges (configuration parameter) for a percentage r of clients. Specifically, for these clients, we randomly flip, remove or add edges to their DAGs while preserving acyclicity. The resulting aggregated $\mathcal{G}^{(k)}$ is the graph provided to the server. Local datasets are then obtained by partitioning $\mathcal{D}$ into $\mathcal{K}$ disjoint subsets of approximately equal size. For client k, we retain only the supervision available in its subgraph, i.e., for the concepts and/or the task contained in the set of nodes of the subgraph. All other concept and task are treated as unobserved for that client.

C.2 Training details

Optimization and initialization. All models are trained using the Adam optimizer with LeakyReLU activations throughout. Experiments are repeated over different random seeds, affecting data partitioning, client assignment, and model initialization. Newly instantiated layers, including those added during architecture adaptation, use PyTorch's default random initialization, whose weights and biases are zero-centred in expectation and reproducible for each seed. The batch size is fixed to 512 across all datasets and learning regimes. All models use the loss defined in Eq. (1), with the concept–task trade-off parameter set to $\gamma = 0.8$. Key hyperparameters—including learning rate, MLP hidden dimension, and dropout rate—are tuned via grid search on validation sets independently for each dataset and model. Complete hyperparameter configurations for all methods and training regimes are provided in the YAML configuration files released with the code. During training, we apply random concept interventions with probability 0.8, following the intervention protocol of Espinosa Zarlenga et al. [6], which encourages robustness to concept-level corrections.

Centralized and localized training. For centralized and localized baselines, models are trained for up to 200 epochs with early stopping based on the validation loss, using a patience of 10 epochs without improvement. To ensure a fair comparison with federated methods, we control for the total amount of training data. In the centralized setting, we train on the union of all data available across the federation over time, i.e., pooling the datasets of all clients that appear in $\mathcal{K}(t)$ throughout training. In the localized setting, each model is trained using data from a single client; the number of localized models equals the size of the largest client pool.

Federated training. Federated methods are trained for up to 200 communication rounds. At each round, clients perform 2 local epochs before sending updates to the server. Early stopping is applied based on the global validation loss, with a patience of 10 rounds without improvement. Unless otherwise specified (e.g., when varying the number of clients in Section D.7.1), the number of participating clients per round is fixed to $|\mathcal{K}_t| = 10$ across all experiments. All federated experiments are conducted under *statistical heterogeneity* and *temporal non-stationarity* (thus resulting in data drift). The client pool $\mathcal{K}(t)$ evolves over time, reaching a total of 20 clients. After an initial warm-up phase—at round $t = 10$ for ASIA, SACHS, and SIIM, and at $t = 20$ for the remaining datasets—new clients join the federation. These late-arriving clients introduce additional concept supervision and, for graph-based instantiations, potentially novel dependency structures. Following client introduction,

the client pool is balanced such that approximately 50% of clients belong to the initial data distribution and 50% to the new distribution.

Parameter freezing under partial supervision. To enforce *module-specific training* under partial concept/task supervision (Section 3.2.3), each client updates only the modules it can directly supervise. Concretely, concept (and task) labels may be missing at a client, which we represent via a missing-label mask (e.g., value -1). At the start of local training for client k, we identify the supervised variable set $\mathcal{V}^{(k)}$. We then *freeze* all concept modules $h_{\phi,j}$ and task module f_ψ of the variables not contained in $\mathcal{V}^{(k)}$. This mechanism applies uniformly across instantiations (CBM/CEM/C^{2}BM/CGM): regardless of parameterization, any module without local supervision is held fixed during the client update, while only supervised modules (and the shared encoder) are optimized. As a result, local updates match Eq. (1) by preventing gradients from modifying unsupervised components, and naturally supports module-wise aggregation since clients return updates only for the modules they trained.

Proxy DAG construction and cycle resolution. When a ground-truth DAG is available, we use it directly as the reference global structure. Otherwise, following the general strategy of De Felice et al. [19], we construct a *proxy* causal graph from the available concepts and task variable by combining data-driven structure discovery with background knowledge. In particular, we first apply causal discovery to obtain an initial directed graph from the observational data, and then refine it with a retrieval-augmented LLM using domain knowledge sources to improve edge plausibility and orientation. This produces a single global proxy DAG, which is then used as the reference structure for centralized training and as the source from which client-specific partial views are derived. In the federated setting, each client is assumed to observe only an imperfect subgraph of this global structure. The server aggregates the client-provided edges into a single graph considering weights proportional to client dataset sizes and then applies a lightweight post-processing step to ensure acyclicity. Concretely, we run a standard depth-first search to detect back-edges and remove them; this procedure is repeated iteratively until no directed cycle remains. Since graph updates occur only when the shared concept set or structural proposals change, this cycle-resolution step is performed infrequently and adds negligible overhead.

C.3 Licenses and hardware

All experiments were implemented in Python using open-source libraries. The core implementation relies on PyTorch 2.12 [65] (BSD-3-Clause) for model training, Flower 1.29 [66] (Apache 2.0) for the federated learning pipeline, and PyTorch Lightning 2.6 (Apache 2.0) for training orchestration. We further used NumPy 2.4 (BSD-3-Clause), SciPy 1.16 (BSD-3-Clause), Pandas 2.3 [67] (BSD-3-Clause), and scikit-learn 1.8 [68] (BSD-3-Clause) for data processing and evaluation, and Matplotlib 3.10 [69] (Matplotlib license, BSD-compatible) for visualization. To support the concept and graph-based components of our methodology, we used bnlearn 0.12 (MIT), pgmpy 0.1 (MIT), and causal-learn 0.1.4 (MIT). For the medical imaging experiments, we additionally relied on TorchXRayVision 1.4 (Apache 2.0). The complete codebase and instructions for reproducing all experiments are available on GitHub[5] under the MIT license.

We use publicly available datasets and pretrained models under their respective licenses and terms of use. Specifically, we use the ASIA, SACHS, ALARM, INSURANCE, and HAILFINDER networks from the bnlearn repository [32], SIIM-PNEUMOTHORAX under the applicable Kaggle competition terms, and CHEXPERT [40] and CHEXPERT PLUS [59] under their respective data-access and research-use agreements. Where an explicit open-source license is unavailable, we follow the dataset provider's access and usage conditions. The pretrained CXR-CLIP [38] and CXR-BERT [60] models are used according to their respective licenses. All datasets and pretrained models are credited through their original references in the paper.

Experiments were run on a server with four NVIDIA RTX A6000 GPUs (48 GB each), dual AMD EPYC 7513 32-core CPUs, and 512 GB RAM.

[5] https://github.com/francescoTheSantis/F-CMs

D Additional experiments

D.1 Concept accuracy

In Table 3, we report concept accuracy as the average accuracy across all concepts (no downstream task), evaluated in the same experimental setting as Section 4.1. However, since concept coverage has already been reported in Section 4.1, we report here, for each benchmark, model, and training regime, the concept accuracy computed only over the concepts predicted in each configuration.

Across benchmarks, models, and training regimes, concept accuracy is generally comparable, with minor differences observed in some localized and static baselines. This is expected, since for each configuration the metric is computed only over the concepts predicted by that configuration. Thus, differences in concept coverage and downstream task performance are not reflected in this measure.

Table 3: Concept Accuracy.

Setting	Model	ASIA	SACHS	ALARM	INSURANCE	HAILFINDER	SIIM PN.	CHEXPERT
Cent. (Upper Bound)	OpaqNN	91.6 ± 0.4	75.3 ±0.5	91.3 ± 0.1	79.5 ± 0.2	61.4 ± 0.3	74.1 ± 0.2	89.4±0.2
	CBM	91.5 ± 0.4	74.8 ± 0.7	91.0 ± 0.1	79.7 ± 0.2	69.7 ± 0.1	74.1 ± 0.1	89.1±0.3
	CEM	91.3 ± 0.4	73.9 ± 0.5	89.9 ± 0.3	77.1 ± 0.3	66.3 ± 0.3	72.8 ± 0.1	89.0±0.1
	CGM	91.0 ± 0.5	72.3 ± 0.5	89.5 ± 0.2	72.7 ± 0.3	63.8 ± 0.3	66.8 ± 0.8	89.2±0.1
	C^2BM	91.3 ± 0.4	74.4 ± 0.5	90.1 ± 0.4	77.5 ± 0.2	65.4 ± 0.6	72.8 ± 0.2	89.1±0.1
Loc.	OpaqNN	87.0 ± 3.4	67.9 ± 0.5	85.3 ± 1.3	73.3 ± 1.2	53.0 ± 1.8	67.2 ± 0.6	87.5±1.4
	CBM	86.8 ± 3.8	68.7 ± 0.6	87.3 ± 0.9	74.4 ± 1.1	60.3 ± 1.0	66.1 ± 0.5	88.5±1.4
	CEM	84.5 ± 4.8	64.4 ± 1.6	85.0 ± 1.3	73.6 ± 1.0	52.3 ± 3.7	67.1 ± 1.4	87.5±1.4
	CGM	85.3 ± 4.4	62.0 ± 2.0	81.7 ± 1.6	64.0 ± 0.8	50.2 ± 1.7	61.7 ± 0.8	88.6±1.3
	C^2BM	86.5 ± 3.8	63.2 ± 1.7	81.9 ± 1.2	70.6 ± 0.9	48.7 ± 3.2	64.5 ± 1.7	87.5±1.4
External	FCL	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Static FL	FedCBM	65.9 ± 4.2	63.0 ± 0.7	62.7 ± 0.2	66.9 ± 0.6	60.2 ± 0.2	63.6 ± 0.3	86.0±5.0
S-F-CMs	OpaqNN	89.5 ± 2.8	73.8 ± 0.7	89.9 ± 0.4	76.8 ± 0.9	56.8 ± 0.7	62.4 ± 0.6	86.2±1.1
	CBM	89.8 ± 2.8	74.5 ± 0.5	90.8 ± 0.3	78.3 ± 1.0	67.0 ± 0.9	64.9 ± 1.4	86.5±1.1
	CEM	89.2 ± 2.9	74.0 ± 0.4	89.6 ± 0.8	76.2 ± 1.1	60.7 ± 1.2	64.8 ± 0.9	86.0±1.2
	CGM	88.7 ± 2.9	72.3 ± 0.5	88.6 ± 1.4	71.3 ± 1.0	56.6 ± 0.6	62.4 ± 0.9	86.8±1.5
	C^2BM	89.5 ± 2.8	73.2 ± 0.8	88.3 ± 1.3	75.2 ± 1.0	60.0 ± 0.7	64.4 ± 1.0	85.7±1.2
F-CMs	OpaqNN	91.0 ± 0.5	72.8 ± 0.6	90.5 ± 0.2	77.0 ± 0.4	56.2 ± 0.3	64.8 ± 0.5	89.1±0.1
	CBM	91.2 ± 0.4	74.2 ±0.6	90.9 ± 0.3	78.9 ± 0.5	66.3 ± 0.3	68.7 ± 0.2	89.2±0.1
	CEM	90.2 ± 1.0	72.9 ± 0.8	90.2 ± 0.3	77.2 ± 0.4	60.4 ± 0.3	66.4 ± 0.8	89.1±0.1
	CGM	88.7 ± 0.7	68.8 ± 0.5	88.9 ± 0.6	72.3 ± 0.7	55.5 ± 0.4	61.9 ± 1.2	89.2±0.2
	C^2BM	90.7 ± 0.5	71.3 ± 0.8	90.0 ± 0.2	75.8 ± 0.9	59.6 ± 0.5	66.3 ± 1.6	89.1±0.1

D.2 Concept-level interventions - further experiments

In this section, we complement the concept-level intervention analysis presented in Sec. 4.2. Specifically, under the same experimental setting, we report: (i) the *task accuracy* under interventions, instead of label accuracy, of the same models, training regimes, and benchmarks evaluated in Sec. 4.2 (App. D.2.1); (ii) the *task* and *label accuracy* under interventions across models and benchmarks different from the ones evaluated in Sec. 4.2 (App. D.2.2); and (iii) a *comparison*, in terms of task and label accuracy under interventions, of the different *F-CM instantiations* (App. D.2.3).

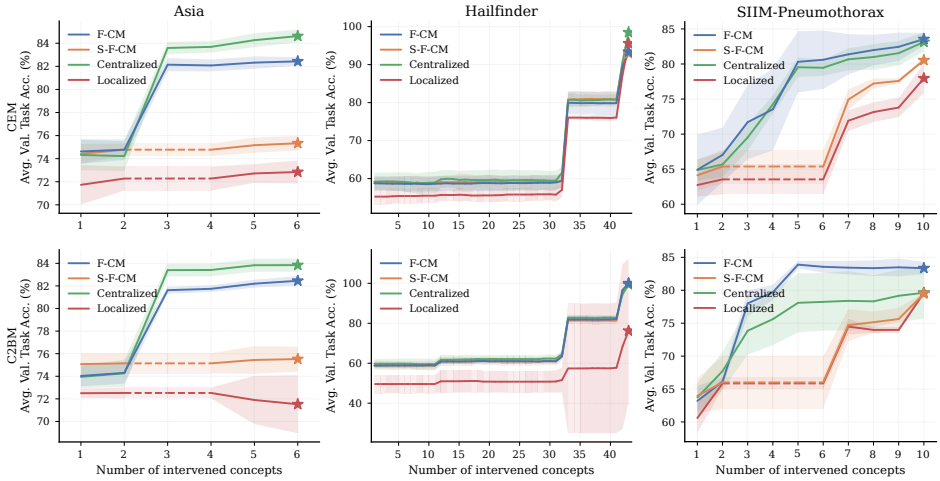


Figure 4: **Task accuracy** (%) following interventions on concepts at increasing depths of the ground-truth graph hierarchy. Dashed lines denote interventions that are not possible for a given training setting due to the absence of predictions for the corresponding concept.

D.2.1 Task accuracy

In Fig. 4, we report the *task accuracy* under interventions for the same models, benchmarks, and training regimes considered in Sec. 4.2. The results are consistent with the findings discussed in that section for label accuracy.

D.2.2 Different Models and Benchmarks

In Fig. 5 and Fig. 6, we repeated the experiments of Sec. 4.2 and App. D.2.1 using different model architectures. Specifically, we report both label and task accuracy for the F-CM instantiations not previously shown (CBM and CGM), evaluated under the same experimental setup as in Sec. 4.2. The results are consistent with the findings discussed in that section.

For further reference, we reported the same results for the other real-world dataset considered, CHEXPERT, in Fig. 7 and Fig. 8. Again, results are consistent with the findings discussed in Sec. 4.2.

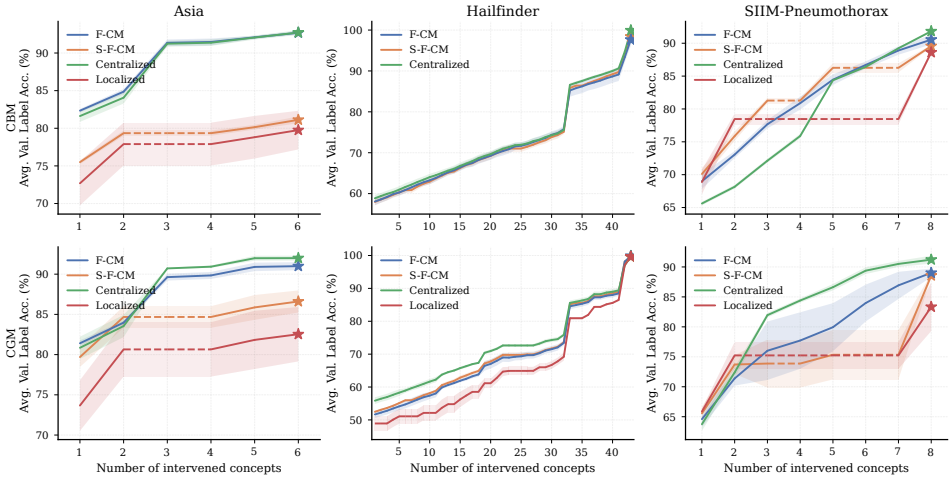


Figure 5: **Label accuracy** (%) on downstream variables (including the task) following interventions on concepts at increasing depths of the ground-truth graph hierarchy. Dashed lines denote interventions that are not possible for a given training setting due to the absence of predictions for the corresponding concept.

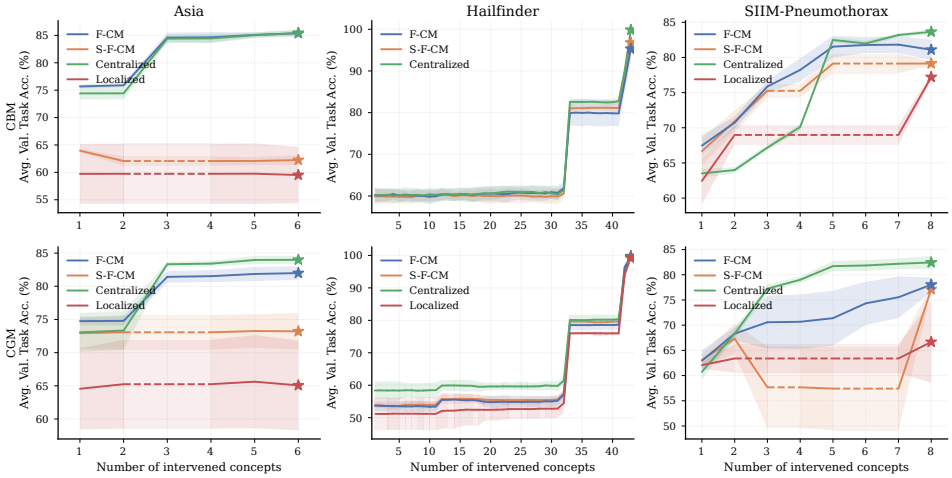


Figure 6: **Task accuracy** (%) following interventions on concepts at increasing depths of the ground-truth graph hierarchy. Dashed lines denote interventions that are not possible for a given training setting due to the absence of predictions for the corresponding concept.

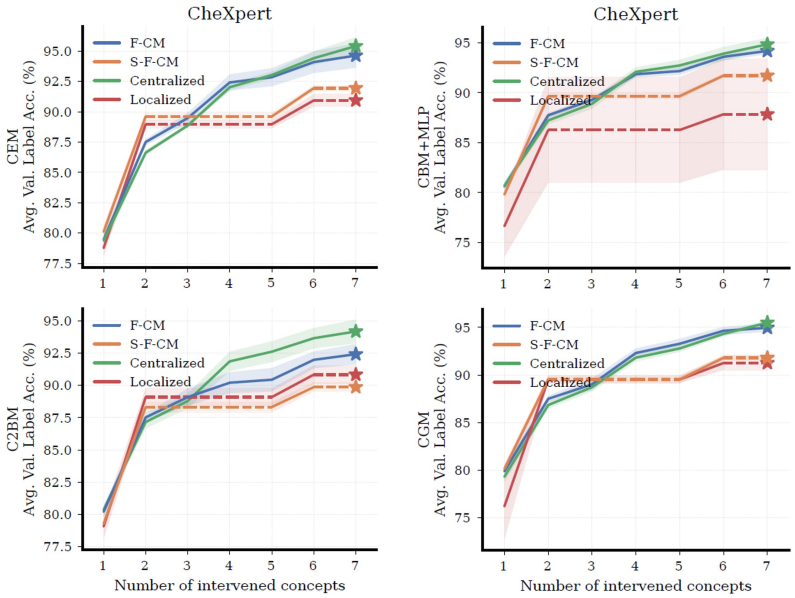


Figure 7: **Label accuracy** (%) on downstream variables (including the task) following interventions on concepts at increasing depths of the ground-truth graph hierarchy. Dashed lines denote interventions that are not possible for a given training setting due to the absence of predictions for the corresponding concept.

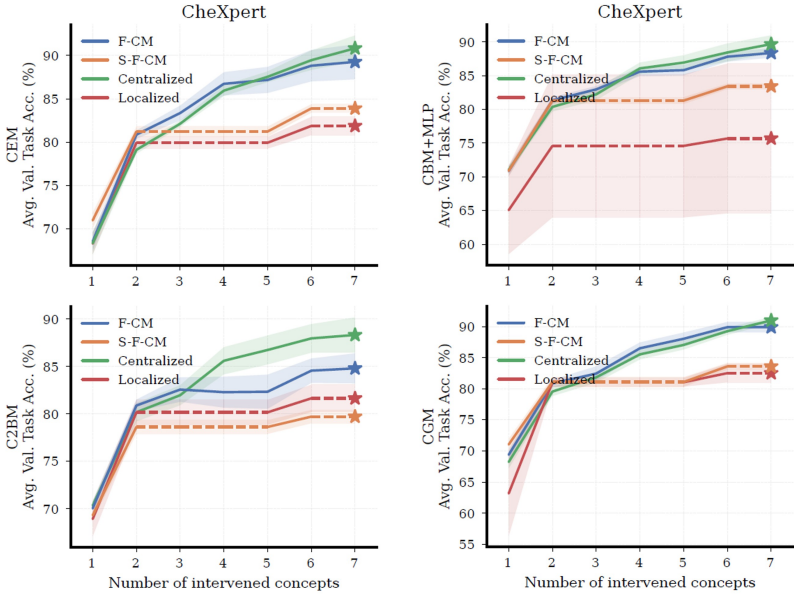


Figure 8: **Task accuracy** (%) following interventions on concepts at increasing depths of the ground-truth graph hierarchy. Dashed lines denote interventions that are not possible for a given training setting due to the absence of predictions for the corresponding concept.

D.2.3 Comparison of F-CMs Instantiations

We further compare how different CM instantiations benefit from the proposed federated protocol by evaluating performance under concept-level interventions. Figures 9, 10 and 11 report the *label accuracy* and *task accuracy* obtained after intervening on concepts sampled at increasing depths of the ground-truth hierarchy (Sec. 4.2). Overall, both metrics exhibit consistent trends: improvements in corrected concept predictions translate into accuracy gains, with larger gains typically obtained when intervening on upstream concepts.

On ASIA, the bipartite CBM variant often shows the strongest response to interventions. We attribute this to the small scale and limited structural complexity of ASIA: when the concept set is compact and most concepts have short causal paths to the task, explicitly modeling concept dependencies offers limited additional benefit, while the simpler bipartite parameterization can be easier to optimize and less sensitive to graph misspecification. On the larger and more structurally rich datasets (HAILFINDER, SIIM-PNEUMOTHORAX and CHEXPERT), graph-based instantiations tend to yield stronger gains under interventions, followed closely by CEMs, which have higher representational flexibility than CBMs due to their ability to encode richer concept representations.

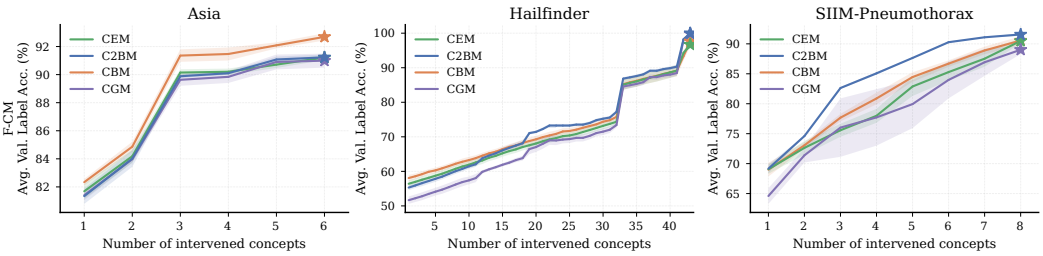


Figure 9: **Label accuracy** (%) following interventions on concepts at increasing depths of the ground-truth graph hierarchy across CMs instantiations.

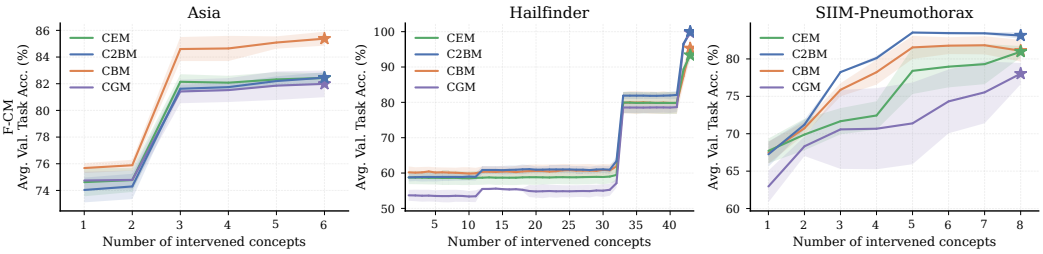


Figure 10: **Task accuracy** (%) following interventions on concepts at increasing depths of the ground-truth graph hierarchy across CMs instantiations.

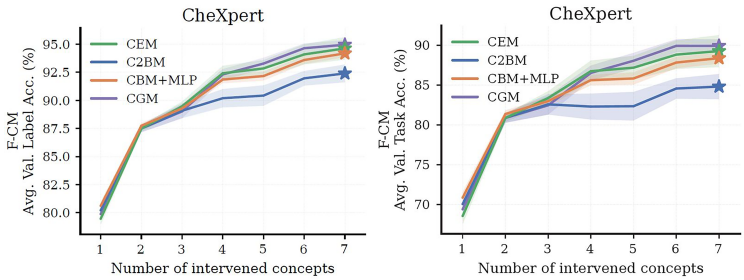


Figure 11: **Label** and **task accuracy** (%) on CHEXPERT following interventions on concepts at increasing depths of the ground-truth graph hierarchy across CMs instantiations.

D.3 Additional model instances for efficient adaptation in non-stationary federations

To complement the analysis in Sec. 4.3, we report the same non-stationary federation experiment for two additional model instantiations, CBM and CGM. As in the main paper, we compare **F-CMs** (ours), **S-F-CMs** (ours without architecture adaptation), and **S-F-CMs Reinit** (full retraining after federation growth), and measure the average validation task loss across communication rounds when newly joined clients introduce previously unseen concepts.

Fig. 12 shows trends that are qualitatively consistent with those reported in the main paper for C^{2}BM and CEM. In both additional instantiations, **F-CMs again adapts more efficiently to federation growth**, reaching lower loss on average than the static baseline and avoiding the slow recovery induced by re-initialization from scratch. Overall, these additional results further support the conclusion that selectively expanding only the modules affected by newly introduced concepts or dependencies yields a substantially more efficient adaptation strategy than either freezing the architecture or fully re-training the model after each structural change.

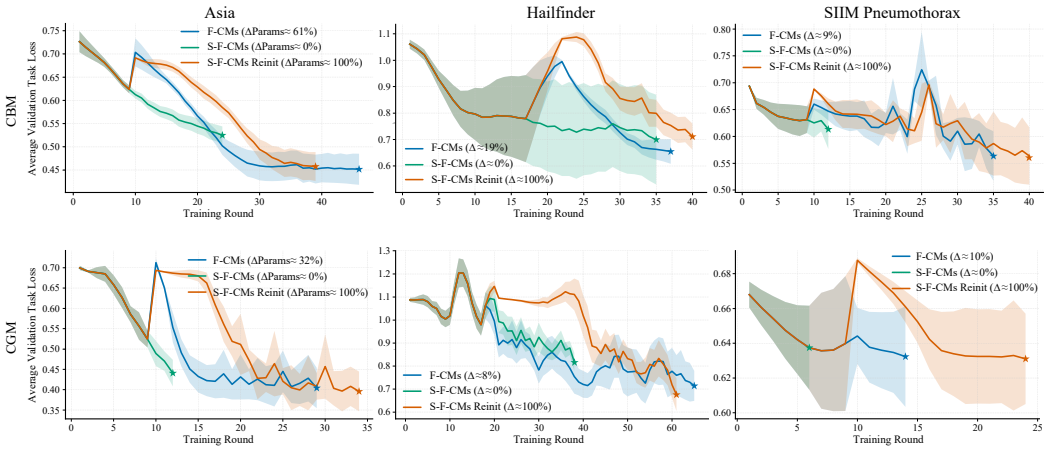


Figure 12: **Convergence under temporal non-stationarity for additional model instantiations.** Average validation task loss across rounds for CBM and CGM, where new clients introduce previously unseen concepts. F-CMs (ours) adapts the architecture modularly and converges faster and to lower loss than S-F-CMs (no adaptation) and S-F-CMs Reinit (full retraining). Curves are truncated at the best loss-round; the omitted tail corresponds to no-improvement rounds. The legend reports the fraction of modified parameters.

D.4 Extent of architecture updates under temporal non-stationarity

To quantify the efficiency of *dynamic architecture adaptation* under temporal non-stationarity, we measure the fraction of shared model parameters whose values change after new clients introduce previously unseen concepts (and, for graph-based instantiations, updated dependencies). Concretely, we report the percentage of parameters that differ between the shared model before the arrival of new clients and the final shared model at the end of training, averaged across seeds. We use this metric as a proxy for the *scope of adaptation* and *training compute-efficiency*: smaller values indicate that updates remain localized to a limited set of concept/task modules rather than inducing widespread changes.

Table 4 reports results across datasets and model instantiations. Overall, the fraction of changed parameters remains far below full retraining (i.e., $\ll 100\%$), supporting our main claim that F-CMs incorporate new supervision by updating only the affected modules while preserving the rest of the model. The magnitude of the update varies across architectures and datasets, reflecting differences in concept dimensionality and the degree to which newly introduced supervision interacts with existing modules (e.g., CEM typically changes a larger fraction than C^{2}BM). These results complement the convergence trends in Fig. 3: sparser updates are associated with faster recovery after client arrivals, avoiding the slow restart behaviour of retraining from scratch.

Table 4: **Fraction of parameters changed (%)** due to *temporal non-stationarity*.

Model	ASIA	SACHS	ALARM	INSURANCE	HAILFINDER	SIIM PN.
OpaqNN	18.28±2.56	38.32±6.40	42.76±5.82	28.90±4.24	17.37±2.93	5.46±0.67
CBM	62.55±12.93	70.00±15.48	48.44±7.88	26.90±3.58	19.24±4.21	9.27±1.26
CEM	61.30±11.83	81.11±18.54	57.30±8.84	30.44±4.48	17.80±3.13	21.61±3.01
CGM	32.19±3.17	43.09±7.05	45.87±10.04	19.68±3.39	8.37±1.45	10.29±1.33
C^2BM	28.56±4.54	44.82±8.49	31.06±4.62	19.24±2.78	12.73±2.05	8.49±1.13

D.5 Computational overhead of graph aggregation and modular adaptation

We analyze the computational implications of F-CMs from two complementary perspectives: the additional overhead introduced by graph aggregation and modular architecture adaptation, and the scaling of the shared model as new concepts accumulate. The former comes from two main sources: *(i)* server-side graph aggregation and cycle resolution, and *(ii)* sparse architecture adaptation when newly arrived clients introduce unseen concepts or dependencies.

Complexity analysis. Consider a client k supervising m_k variables. If the client communicates a dense local adjacency/confidence matrix, the structural communication cost is $\mathcal{O}(m_k^2)$, and aggregating the client-provided structures at the server costs $\mathcal{O}\big(\sum_{k\in\mathcal{K}_t} m_k^2\big)$. Importantly, this cost is independent of the number of local training samples, local epochs, and SGD steps, and is therefore small relative to the dominant FL cost, namely repeated forward/backward passes over model parameters. For intuition, even with $m_k = 50$, a dense graph contains only 2500 entries; with float32 values this corresponds to roughly 10 KB of transmitted data, which is negligible compared to communicating model updates for networks with millions of parameters, typically requiring several MBs. Moreover, graph communication is *event-triggered*: a client only needs to transmit its local graph when it first joins the federation, or when its graph changes. Hence, this cost is amortized over rounds, and in most rounds the communication pattern is unchanged from standard FL. After combining the client-provided graphs, the server applies the cycle-resolution step described in App. C.2, which depends only on the number of currently known variables m_t and remains a lightweight server-side operation.

Architecture adaptation is similarly sparse. When the shared structure changes, F-CMs keep the encoder fixed and expands or rewires only the concept/task modules whose interfaces are affected, while warm-starting all unchanged parameters. Therefore, the adaptation cost scales as $\mathcal{O}(\Delta P_t)$, where ΔP_t is the number of modified parameters, rather than $\mathcal{O}(P_t)$ for full retraining. This is consistent with the results in Table 4: in our experiments, the fraction $\Delta P_t/P_t$ remains well below full reinitialization, explaining the faster recovery observed in Sec. 4.3 and App. D.3.

For ordinary FL communication, F-CMs already performs module-specific local training and module-wise aggregation, so only supervised modules produce updates. As a result, the uplink communication is proportional to the updated subset rather than the full model. The downlink can remain identical to standard FL when the whole model is broadcast. Overall, compared to static FL, F-CMs introduce only a modest and mostly event-triggered structural overhead while gaining the ability to absorb late-arriving concepts; compared to retraining from scratch, it is substantially cheaper because it avoids re-optimizing the full model.

Runtime overhead. Table 5 reports the average runtime of the main additional operations introduced by F-CMs. The results confirm that these costs are negligible in practice compared to the overall training time. In particular, graph aggregation requires only a few hundredths of a second when present, while post-drift architecture instantiation remains below one second across all reported settings. Parameter transfer from the pre-drift to the post-drift model is even smaller. By contrast, the total end-to-end training time is in the order of hundreds of seconds, showing that the computational overhead of structure handling and modular adaptation is minimal relative to the dominant cost of repeated local training and communication rounds.

Scaling under concept accumulation. As defined in Sec. 2.3, F-CMs retain concepts already incorporated into the shared model when clients leave, since client absence does not necessarily imply that the corresponding concepts are obsolete. This conservative choice avoids prematurely

Table 5: **Average runtime (seconds) of the additional operations introduced by F-CMs**, reported as mean $\pm$ 95% confidence interval over three runs. Here, $d_inst.$ denotes the time spent instantiating the architecture after client drift, $d_par.$ the time spent initializing the post-drift architecture with previously learned parameters, $agg.$ the time spent aggregating client graphs, n_rounds the number of training rounds, and $tot.$ the total training time. The results show that graph aggregation and modular adaptation contribute only a very small fraction of the total runtime.

Data	Model	$d_inst.$	$d_par.$	$agg.$	n_rounds	$tot.$
ASIA	F-CMs (CEM)	0.227 ± 0.001	0.002 ± 0.000	–	42.583 ± 2.256	229.726 ± 13.609
SIIM-PN.	F-CMs (CEM)	0.691 ± 0.093	0.003 ± 0.000	–	35.350 ± 5.188	506.230 ± 75.622
ASIA	F-CMs (C^2BM)	0.244 ± 0.001	0.003 ± 0.000	0.063 ± 0.051	61.500 ± 3.679	370.800 ± 24.924
SIIM-PN.	F-CMs (C^2BM)	0.672 ± 0.102	0.004 ± 0.000	0.037 ± 0.003	56.550 ± 7.278	922.543 ± 120.707

Table 6: **Scaling under concept accumulation.** Growth of the concept-dependent component for CEM and C^2BM as new concepts are incorporated. "New params." denotes the parameters of a newly instantiated concept module, while "Added params." also includes affected interfaces. Memory is the post-drift FP32 parameter footprint. The fixed input encoder is excluded from all measurements.

		New params./concept		Added params.		Memory (MiB)		Forward MAC increase	
Dataset	Concepts added	CEM	C^2BM	CEM	C^2BM	CEM	C^2BM	CEM	C^2BM
ASIA	3	1,346	610	5,130	1,982	0.066	0.044	42.0%	20.7%
ALARM	8	3,843	1,763	48,513	15,516	0.745	0.335	33.1%	21.4%

discarding potentially relevant knowledge.[6] We therefore additionally quantify how the shared CM scales as new concepts accumulate. Specifically, we measure the parameters introduced by each new concept module, the total parameter growth including affected interfaces, the resulting FP32 memory footprint, and the increase in forward-pass multiply–accumulate operations (MACs). We report results for CEM and C^2BM on ASIA and ALARM. All quantities isolate the concept-dependent component of the model; the fixed input encoder is excluded, as its size and computation do not depend on the number of concepts.

Table 6 shows that model growth remains modest in absolute memory in the evaluated settings: the largest post-drift concept-dependent component occupies 0.745 MiB in FP32. The parameter cost of an individual new concept module ranges from 610 to $3{,}843$ parameters, with additional growth arising when existing concept or task interfaces are affected. Forward computation increases more visibly, by 20.7–42.0% for the concept-dependent component, although the corresponding full-model increase is lower because the fixed input encoder is unchanged. Overall, the scaling cost is architecture-dependent, with C^2BM exhibiting lower parameter and forward-computation growth than CEM in both evaluated settings.

Importantly, concept accumulation does not induce a proportional increase in client-side backward computation or uplink cost, since unsupervised modules remain frozen and are not transmitted; the measured forward-cost growth instead reflects that the CM instantiations considered here construct the complete concept representation for task prediction.

D.5.1 Ablation of Interface-Initialization Strategies

Dynamic architecture adaptation changes the interface of the affected modules. Although F-CMs localize each structural update and preserve all unaffected modules, reinitializing a previously learned interface may temporarily perturb optimization. We therefore investigate how different initialization strategies affect the short-term adaptation dynamics and the final validation loss after concept-space expansion. We compare three strategies that differ only in how the modified interface is initialized:

1. **Current strategy (Current).** The resized input or output interface of the affected module is reinitialized, while all unaffected modules retain their parameters. See more details in Section 3.2.2 and App. B.

[6] The modular adaptation mechanism is nevertheless compatible with concept removal: the corresponding module could be removed, the shared DAG updated as for other structural changes, and only the affected interfaces adapted. Developing principled criteria for concept pruning while preserving relevant knowledge is discussed as a future research direction.

2. **Partial reuse with random initialization (Partial-random).** Parameters associated with existing inputs or outputs are copied from the pre-update module, while only those associated with newly introduced inputs or outputs are randomly initialized. For example, suppose the task decoder is initially trained to predict Y from C_1 . When a new concept C_2 is added as an input, we expand the first layer of the task decoder to receive (C_1, C_2). The block of weights associated with C_1 is copied from the previously learned decoder, while the new block associated with C_2 is randomly initialized. Training then proceeds as in standard F-CMs.

3. **Partial reuse with zero initialization (Partial-zero).** Parameters associated with existing inputs or outputs are preserved, as in the previous strategy, while the newly introduced parameters are initialized to zero. When applicable, this initialization preserves previously learned dependencies immediately after the architectural change, while allowing new dependencies to be learned during subsequent training.

We evaluate the three strategies using C^2BM and CGM instantiations on the ASIA and ALARM datasets. For each dataset, we consider two concept-space expansion scenarios, from initial coverage ranges of 10–50% and 51–75%, respectively, to full coverage. We quantify coverage using *structural concept coverage*, a model-independent measure of the fraction of nodes in the ground-truth graph covered by the union of the selected client subgraphs (see App. D.7.5 for details).

Figures 13 and 14 show how the validation task loss evolves before and after the structural update under the three interface-initialization strategies. Each figure reports results for C^2BM and CGM under the 10–$50\% \rightarrow 100\%$ and 51–$75\% \rightarrow 100\%$ coverage expansions. At each federated round, the loss is averaged across the validation sets of the clients in the training federation. Curves and shaded regions report the mean $\pm$ standard deviation over three seeds, while the vertical dashed line marks the structural update. To illustrate longer-term training dynamics, we allow the models to train for additional rounds. However, model selection is based on the lowest validation loss achieved during training, rather than the loss at the final round.

On ASIA (Fig. 13), both concept-space expansion scenarios exhibit broadly similar overall trends. Immediately after the structural update, the two partial-reuse strategies generally achieve an earlier reduction in validation loss, while the current strategy exhibits a transient increase, particularly for CGM and C^2BM in the second expansion scenario. These differences diminish during subsequent training, with all three strategies reaching comparable validation losses. In the second expansion scenario for C^2BM, the current strategy attains a slightly lower late-stage loss, although the standard-deviation bands overlap. The trajectories of *partial-random* and *partial-zero* remain close, suggesting that the main short-term benefit comes from preserving existing parameters rather than from the initialization of newly introduced ones.

On ALARM (Fig. 14), the two concept-space expansion scenarios exhibit broadly similar trends. For C^2BM, the partial-reuse strategies achieve a faster initial reduction in validation loss after the structural update, while the current strategy adapts more gradually. This difference is particularly evident in the second expansion scenario, where the current strategy exhibits a transient increase in loss. For CGM, the three strategies show largely comparable trajectories, with only minor differences during post-update adaptation.

In later training rounds, the current strategy reaches a numerically lower validation loss for C^2BM in both expansion scenarios, although the standard-deviation bands overlap. For CGM, the strategies achieve comparable losses without a consistent advantage for any initialization approach.

Taken together, the results on ASIA and ALARM suggest that preserving parameters associated with the existing interface generally facilitates the initial adaptation following a structural update. However, this short-term benefit does not consistently translate into lower validation loss later in training. The current strategy typically catches up with the partial-reuse variants and can achieve comparable or, in some settings, numerically lower losses. Moreover, the relative differences between strategies do not follow a consistent pattern across the two expansion scenarios. Overall, no strategy consistently outperforms the others across datasets, architectures, and concept-space expansions.

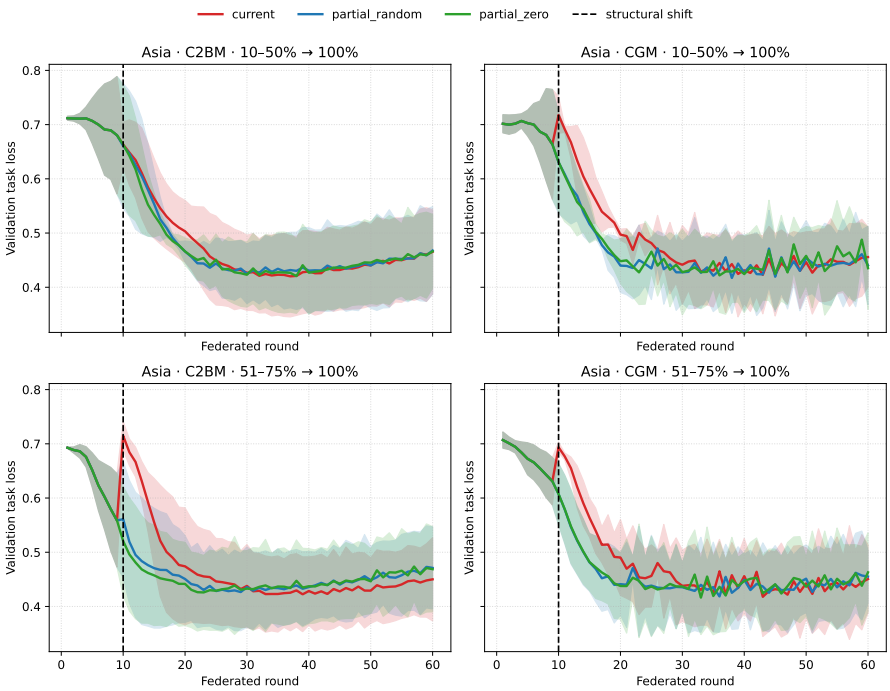


Figure 13: **Interface-initialization ablation on ASIA.** Average validation task loss across federated rounds for C^{2}BM (left) and CGM (right), under concept-space expansions from 10–50% to 100% structural concept coverage (top) and from 51–75% to 100% coverage (bottom). We compare the current initialization strategy with partial parameter reuse, using either random or zero initialization for newly introduced parameters. The vertical dashed line marks the structural update. Curves and shaded regions report mean ± standard deviation over three seeds.

D.6 Differential privacy

We investigate two complementary privacy mechanisms for F-CMs. First, we consider sample-level differential privacy (DP) for model updates, assessing its impact on predictive and concept performance (App. D.6.1). We then study an optional edge-level differentially private mechanism for client graph communication, evaluating the resulting trade-off between structural privacy, graph quality, and predictive performance (App. D.6.2).

D.6.1 Sample-level privacy of model updates

We first consider sample-level DP for communicated model updates, limiting the information that can be inferred about individual training samples. DP provides a formal guarantee that the inclusion or exclusion of any single data sample has a bounded impact on the released information, thereby protecting individual participants against inference attacks. Formally, a randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies (ε, δ)-differential privacy if, for any pair of neighboring datasets D and D' that differ in one sample and any measurable set $\mathcal{S} \subseteq \mathcal{R}$,

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \;\leq\; e^{\varepsilon} \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta. \tag{2}$$

Here, ε controls the strength of the privacy guarantee, with smaller values corresponding to stronger privacy protection, while δ represents a negligible probability of privacy violation.

In practice, we enforce DP at the client update level using a standard DP-SGD pipeline implemented with Opacus [70]. At each local optimization step, client gradients are computed at the *per-sample* level and then *clipped* to a maximum ℓ_2 norm C:

$$g_i \leftarrow g_i \cdot \min\Big(1, \frac{C}{\|g_i\|_2}\Big), \qquad C = 1.0. \tag{3}$$

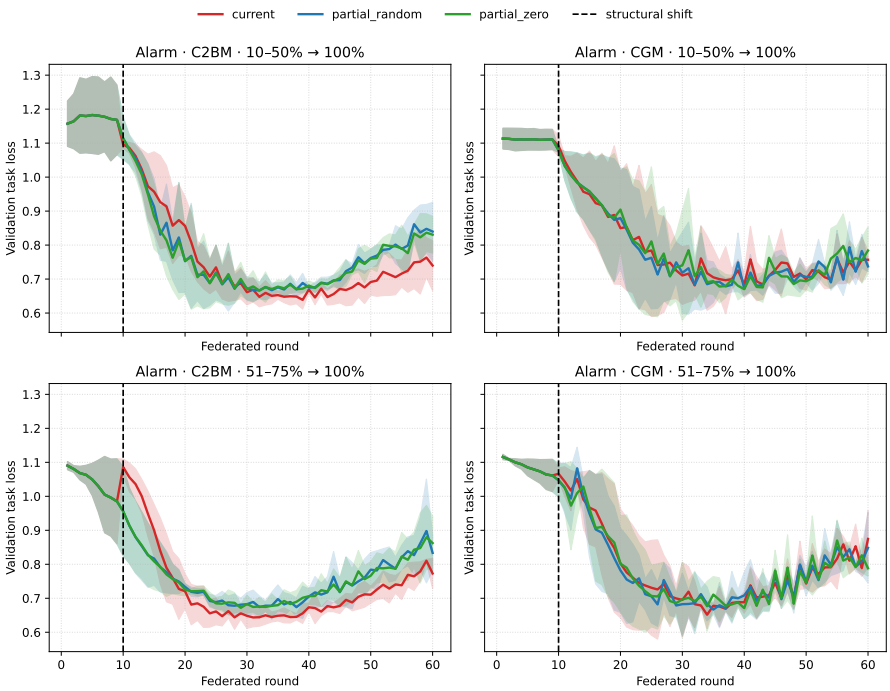


Figure 14: **Interface-initialization ablation on ALARM.** Average validation task loss across federated rounds for C^{2}BM (left) and CGM (right), under concept-space expansions from 10–50% to 100% structural concept coverage (top) and from 51–75% to 100% coverage (bottom). We compare the current initialization strategy with partial parameter reuse, using either random or zero initialization for newly introduced parameters. The vertical dashed line marks the structural update. Curves and shaded regions report mean ± standard deviation over three seeds.

Clipping bounds the contribution of any single sample to the update, which effectively bounds the sensitivity of the released (aggregated) gradient signal. After clipping, Gaussian noise is added to the (mini-batch) aggregated gradient before taking the optimizer step. The noise scale is chosen by the privacy accountant to meet a target privacy budget (ε, δ) given the training configuration (sampling rate, number of steps/epochs, etc.). In our experiments we fix $\delta = 10^{-5}$ and report results for $\varepsilon \in \{10, 5, 1\}$, where smaller ε corresponds to stronger privacy (and typically lower utility).

Empirical Validation. Table 7 reports task, and concept accuracy under DP for different privacy budgets on ASIA. As expected, tightening privacy (decreasing ε) leads to a moderate degradation in performance. Notably, across most instantiations the drop from $\varepsilon = 10$ to $\varepsilon = 5$ is small, and even at $\varepsilon = 1$ several models remain competitive (e.g., OpaqNN and CEM exhibit limited degradation), indicating that F-CM-style training remains effective under strong privacy constraints. Larger drops for some architectures (e.g., CBM and CGM at $\varepsilon = 1$) are consistent with the stronger noise required at low ε and the known sensitivity of some training dynamics to DP perturbations.

Table 7: **Impact of differential privacy on federated concept-based models under different privacy budgets ε.** We report task accuracy and concept accuracy (mean ± std.) for representative model instantiations.

Privacy Budget	$\varepsilon = 10$		$\varepsilon = 5$		$\varepsilon = 1$	
	Task Acc.	**Concept Acc.**	**Task Acc.**	**Concept Acc.**	**Task Acc.**	**Concept Acc.**
OpaqNN	80.03 ± 0.68	90.37 ± 0.70	79.19 ± 0.97	89.43 ± 1.32	77.31 ± 0.96	89.15 ± 1.07
C^{2}BM	79.37 ± 1.01	86.89 ± 2.78	79.20 ± 1.16	86.89 ± 2.63	76.22 ± 2.24	85.81 ± 2.06
CBM	79.11 ± 1.79	89.32 ± 1.46	79.03 ± 1.87	89.13 ± 1.49	72.73 ± 6.52	88.07 ± 2.16
CEM	80.19 ± 1.08	86.26 ± 2.72	80.17 ± 1.63	87.26 ± 2.74	78.12 ± 2.68	85.67 ± 2.47
CGM	78.67 ± 2.39	88.59 ± 1.34	77.68 ± 0.94	86.47 ± 1.90	72.00 ± 2.23	85.83 ± 1.83

Table 8: **Edge-level structural privacy.** Task accuracy (%) for C^{2}BM and CGM and graph disagreement (GD, %) with respect to the non-private aggregate under different structural privacy budgets ϵ_s.

Dataset	ϵ_s	C^{2}BM	CGM	GD (%)
ASIA	∞	80.4 ± 2.3	77.0 ± 3.9	0
	8	80.4 ± 2.3	77.0 ± 3.9	0
	4	80.4 ± 2.3	77.0 ± 3.9	0
	2	80.2 ± 1.9	77.9 ± 1.7	3.6 ± 3.6
	1	79.8 ± 2.4	77.8 ± 1.5	21.4 ± 11.3
ALARM	∞	72.6 ± 0.9	71.1 ± 0.6	0
	8	72.6 ± 0.9	71.0 ± 0.7	0
	4	72.5 ± 2.7	70.9 ± 0.5	1.0 ± 0.5
	2	72.8 ± 2.2	65.5 ± 8.6	6.6 ± 1.7
	1	70.8 ± 0.4	65.6 ± 10.4	15.8 ± 2.2

D.6.2 Edge-level structural privacy

Complementary to sample-level differential privacy for model updates, we consider structural privacy for client graph communication. In sensitive applications, such structural information may itself reveal properties of a client's data or domain-specific practices. We therefore equip the communicated client structures $A^{(k)}$ with an optional edge-level differentially private mechanism that perturbs pairwise structural relations before transmission.

For each unordered pair $\{V_i, V_j\}$, the client graph assigns one of three possible relations, $\mathcal{R}_{ij} = \{V_i \rightarrow V_j,\ V_j \rightarrow V_i,\ \emptyset\}$. Before transmission, the client independently perturbs the relation associated with each pair using randomized response. Given structural privacy budget ϵ_s, the true relation is retained with probability $p = \frac{e^{\epsilon_s}}{e^{\epsilon_s}+2}$, while each of the other two relations is reported with probability $q = \frac{1}{e^{\epsilon_s}+2}$. For any two possible relations $r, s \in \mathcal{R}_{ij}$ and any reported relation, the ratio between the corresponding reporting probabilities is at most e^{ϵ_s}, providing ϵ_s-edge-level differential privacy for each pair.

This perturbation is compatible with the consensus-based aggregation used by F-CMs. In particular, if f_r denotes the true frequency of relation r across clients, its perturbed frequency satisfies $\mathbb{E}[\widetilde{f}_r] = q + (p-q)f_r$. Consequently, for any two relations r and s, $\mathbb{E}[\widetilde{f}_r] - \mathbb{E}[\widetilde{f}_s] = (p-q)(f_r - f_s)$. Since $p - q > 0$, the ordering of relation frequencies is preserved in expectation.

Empirical evaluation. We evaluate the effect of structural privacy on ASIA and ALARM using two representative graph-based instantiations, C^{2}BM and CGM. In addition to task accuracy, we report graph disagreement (GD), measured relative to the non-private aggregated graph, where lower values indicate greater structural agreement. The non-private setting is denoted by $\epsilon_s = \infty$.

Table 8 shows that moderate structural privacy introduces little degradation. For $\epsilon_s \geq 4$, the aggregated graph remains unchanged on ASIA and differs on average by at most 1.0% on ALARM, while task accuracy is essentially preserved for both architectures. Under stronger perturbation, the effect becomes dataset- and architecture-dependent. At $\epsilon_s = 1$, GD increases to 21.4% on ASIA and 15.8% on ALARM, while C^{2}BM remains comparatively stable, decreasing by 0.6 and 1.8 percentage points, respectively, relative to the non-private setting. CGM is also stable on ASIA, but on ALARM its accuracy decreases by approximately 5.5 percentage points for $\epsilon_s \in \{2, 1\}$. Overall, moderate edge-level privacy preserves both the aggregated structure and predictive performance, whereas stronger structural perturbations may reduce utility.

D.7 Sensitivity analysis

We further assess the stability of F-CMs under key design and deployment choices. In particular, we analyze the effect of (i) varying the number of clients (App. D.7.1), (ii) selective parameter freezing for locally unsupervised modules (App. D.7.2), (iii) the concept-loss weight γ (App. D.7.3), (iv) the timing of client arrivals (App. D.7.4), and (v) the magnitude of concept-space expansion under federation growth (App. D.7.5).

D.7.1 Number of Clients

We study the effect of scaling the number of clients by varying the total client population $K \in \{10, 20, 50, 100\}$, while keeping the overall amount of data fixed. As K increases, each client holds fewer local samples on average, resulting in smaller and more heterogeneous local datasets. This setting reflects realistic large-scale federated deployments, where data are fragmented across many participants.

Tables 9 reports task and concept accuracy for different federation sizes. Across all model instantiations, performance remains stable as the number of clients increases, with only moderate degradation in accuracy for the largest federation sizes. In particular, concept accuracy exhibits limited sensitivity to increasing K, indicating that aggregating concept supervision from a larger number of smaller clients remains effective.

Models relying on more structured dependencies (e.g., C^2BM and CGM) experience slightly larger variance as K grows. This behavior is consistent with the increased statistical heterogeneity and reduced local sample sizes associated with larger client populations. Overall, these results suggest that F-CMs scale gracefully with the number of clients, preserving both concept quality and downstream task performance even in highly fragmented federations.

Table 9: **Scaling with the number of clients.** Task accuracy and concept accuracy (mean $\pm$ std.) as the total number of clients K increases, while keeping the overall amount of data fixed.

# Clients	$K = 10$		$K = 20$		$K = 50$		$K = 100$	
Models	**Task Acc.**	**Concept Acc.**	**Task Acc.**	**Concept Acc.**	**Task Acc.**	**Concept Acc.**	**Task Acc.**	**Concept Acc.**
OpaqNN	80.61 ± 1.98	91.63 ± 0.70	79.83 ± 0.99	90.80 ± 0.40	80.37 ± 1.06	90.03 ± 0.99	79.94 ± 1.15	90.30 ± 0.82
C^2BM	80.14 ± 2.02	91.23 ± 0.81	78.16 ± 3.11	87.81 ± 2.32	80.35 ± 0.72	86.44 ± 2.17	75.38 ± 3.98	86.10 ± 2.42
CBM	79.91 ± 1.98	91.52 ± 0.83	79.25 ± 1.12	91.09 ± 0.61	79.54 ± 1.02	90.94 ± 0.50	78.32 ± 1.33	90.54 ± 0.55
CEM	80.58 ± 2.04	91.40 ± 0.72	80.58 ± 1.11	88.59 ± 1.63	79.92 ± 1.31	87.92 ± 2.01	78.80 ± 1.80	86.19 ± 1.91
CGM	79.33 ± 2.20	91.05 ± 0.89	79.30 ± 1.74	87.76 ± 1.83	78.96 ± 1.77	85.86 ± 2.10	78.84 ± 1.85	86.41 ± 2.14

D.7.2 Effect of Selective Parameter Freezing

We evaluate the empirical impact of *selective parameter freezing*, i.e., freezing concept/task modules that lack local supervision at a client (Sec. 3.2.3). This mechanism enforces module-specific training by ensuring that only supervised components receive gradients, while unsupervised modules remain unchanged. Tables 10–11 show task and concept accuracy with and without freezing for two representative instantiations (C^2BM and CEM). Overall, freezing is largely *performance-neutral*: on smaller graphs such as ASIA, the effect is minimal since only a few modules are unsupervised at each client. However, on larger benchmarks (e.g., ALARM, HAILFINDER, SIIM), where the model contains more concept modules and parent-conditioned dependencies, freezing can slightly improve stability by preventing updates to poorly-supported components and reducing update noise. Beyond accuracy, freezing provides a practical efficiency benefit at the client side, as it reduces the number of trainable parameters during local optimization (lower gradient/optimizer-state footprint and compute), while remaining fully consistent with our module-wise aggregation protocol.

D.7.3 Sensitivity to the concept loss weight γ

We evaluate the sensitivity of our framework to the concept-loss weight γ, which controls the trade-off between downstream task supervision and concept supervision during local optimization. We vary $\gamma \in \{0.2, 0.5, 0.8\}$ and report both task accuracy and concept accuracy for two representative instantiations, C^2BM and CEM, on three representative datasets with different structural and concept-space complexity: ASIA, ALARM, and SIIM-PNEUMOTHORAX (Figure 15).

Overall, the results show that performance is largely robust to the choice of γ. On ASIA and ALARM, both task accuracy and concept accuracy remain nearly unchanged across the tested values for both model instantiations, indicating that the framework is not particularly sensitive to the exact balance between task and concept objectives in these settings. This suggests that the optimization remains stable even when concept supervision is given substantially different weights. A more visible effect appears on SIIM-PNEUMOTHORAX, which has a more challenging concept space. There, increasing γ leads to a mild improvement in concept accuracy, particularly for C^2BM, while task accuracy

Table 10: **Effect of selective parameter freezing (ASIA/SACHS/ALARM).** Task and concept accuracy (mean ± std.) across 5 seeds, comparing standard training (No) vs. selective freezing of unsupervised modules (Yes).

Model	Freezing	ASIA		SACHS		ALARM	
		Task Acc.	Concept Acc.	Task Acc.	Concept Acc.	Task Acc.	Concept Acc.
C^2BM	Yes	80.35 ± 2.46	90.55 ± 1.08	76.20 ± 2.38	72.22 ± 1.91	73.14 ± 0.80	90.23 ± 0.31
	No	80.52 ± 2.24	90.47 ± 1.29	76.55 ± 2.20	72.73 ± 1.59	72.68 ± 1.28	89.12 ± 0.84
CEM	Yes	80.50 ± 1.90	89.77 ± 2.33	75.97 ± 2.92	73.03 ± 1.55	73.42 ± 0.50	90.10 ± 0.63
	No	80.27 ± 1.99	90.07 ± 1.92	76.34 ± 2.58	73.33 ± 1.38	72.79 ± 0.50	89.87 ± 0.71

Table 11: **Effect of selective parameter freezing (INSURANCE/HAILFINDER/SIIM).** Task and concept accuracy (mean ± std.) across 5 seeds, comparing standard training (No) vs. selective freezing of unsupervised modules (Yes).

Model	Freezing	INSURANCE		HAILFINDER		SIIM PN.	
		Task Acc.	Concept Acc.	Task Acc.	Concept Acc.	Task Acc.	Concept Acc.
C^2BM	Yes	74.27 ± 2.67	76.08 ± 1.37	72.83 ± 2.35	57.81 ± 1.61	68.80 ± 2.87	67.68 ± 1.51
	No	74.47 ± 3.01	76.23 ± 1.44	71.70 ± 1.81	56.61 ± 1.45	65.90 ± 2.99	64.91 ± 3.07
CEM	Yes	70.97 ± 0.86	77.16 ± 0.80	70.15 ± 4.36	60.13 ± 1.43	66.33 ± 3.06	64.90 ± 1.80
	No	72.62 ± 2.27	76.92 ± 1.39	70.30 ± 4.24	59.34 ± 1.51	66.91 ± 2.90	65.88 ± 3.24

exhibits a corresponding small decrease. This trend is expected: assigning more weight to the concept loss encourages the model to better align with concept supervision, at the cost of slightly reduced emphasis on the end task. Importantly, even in this more demanding setting, the variation remains moderate, confirming that our method does not rely on careful tuning of γ to achieve strong performance.

D.7.4 Sensitivity to Client Arrival Timing

To assess whether F-CM performance depends on the specific arrival schedule used in the main experiments, we first vary the round at which new clients enter the federation. Specifically, we consider client arrivals at 10%, 30%, 60%, and 90% of the total training duration, as well as a setting with no new client arrivals. We report task accuracy across three random seeds for two representative instantiations, CEM and C^2BM, on the ASIA and ALARM datasets.

Table 12 highlights the dependence of performance on the available adaptation horizon. On ASIA, early and intermediate arrivals ($\leq 60\%$ of the total training duration) leave task accuracy largely unchanged, whereas delaying client arrival to 90% reduces accuracy to 77.8% and 73.8% for CEM and C^2BM, respectively, while still outperforming the no-arrival setting on average. On ALARM, all arrival configurations improve over no arrival, with gains of up to 19 percentage points for C^2BM and 18.5 points for CEM. Overall, these results indicate that F-CMs remain effective across different arrival schedules, while very late arrivals naturally leave less time to incorporate the newly available supervision.

D.7.5 Sensitivity to Concept-Space Expansion

We next study how performance changes as the federation acquires supervision for additional concepts. We consider two expansion transitions: (i) from low to medium federation coverage, corresponding to $< 51\%$ and 51–75% of the concepts in the ground-truth graph, respectively; and (ii) from medium to high coverage, corresponding to 51–75% and $> 75\%$, respectively. We use relative coverage ranges rather than fixed absolute numbers of concepts to make the expansion levels comparable across datasets with different graph sizes.

Table 13 reports the resulting task accuracy for CEM and C^2BM instantiations on ASIA and ALARM, together with the *structural concept coverage*. Unlike the standard concept coverage, this measure is model-independent and quantifies the fraction of nodes in the ground-truth graph covered by the

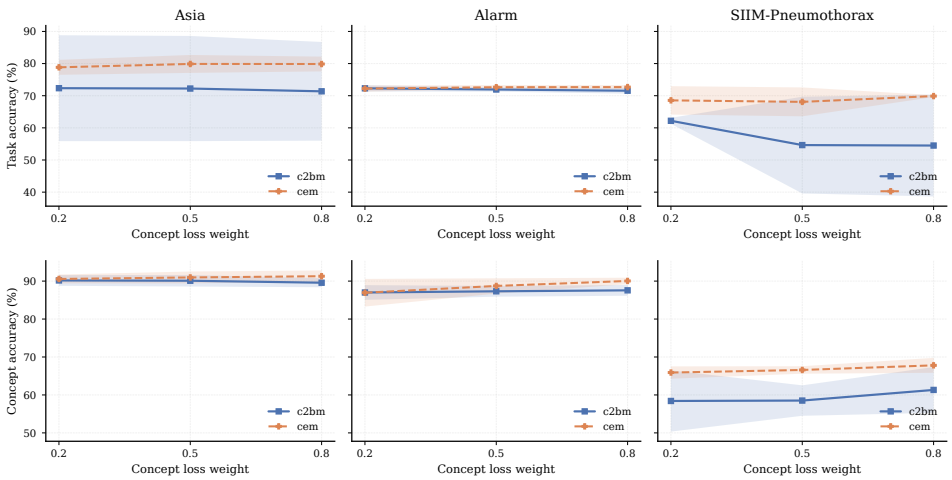


Figure 15: **Sensitivity analysis on the concept loss weight γ.** *Task accuracy* (%, top row) and *concept accuracy* (%, bottom row) as a function of γ for the C^{2}BM and CEM instantiations on ASIA, ALARM, and SIIM-PNEUMOTHORAX, averaged over three runs. Performance is largely stable across values of γ. On SIIM-PNEUMOTHORAX, increasing γ yields a slight improvement in concept accuracy, accompanied by a small decrease in task accuracy, reflecting the expected trade-off between concept fidelity and downstream prediction.

Table 12: **Sensitivity to client arrival timing.** Task accuracy (mean $\pm$ std.) for F-CMs instantiated with CEM and C^{2}BM when new clients arrive after different communication rounds. "No arr." denotes the setting without client arrival. Results are reported over three random seeds.

	ASIA					ALARM				
Model	10%	30%	60%	90%	No arr.	10%	30%	60%	90%	No arr.
CEM	79.3 ± 1.3	79.9 ± 1.2	79.8 ± 0.9	77.8 ± 2.5	67.7 ± 12.9	73.6 ± 0.2	72.6 ± 0.7	70.4 ± 1.4	70.3 ± 1.5	55.1 ± 17.3
C^{2}BM	79.6 ± 1.3	79.8 ± 1.0	79.8 ± 0.9	73.8 ± 6.0	63.5 ± 11.5	73.7 ± 0.4	72.8 ± 0.1	72.5 ± 1.0	71.0 ± 2.3	54.7 ± 17.6

union of the selected client subgraphs. While the coverage ranges define the expansion scenarios, structural concept coverage reports the actual coverage achieved within each range.

Results show an overall positive trend as concept availability increases. On ASIA, increasing concept coverage from 45.8% to 62.5% produces moderate changes in task accuracy, consistent with its smaller causal graph, for which this expansion corresponds to only a few additional concepts. At full coverage, task accuracy increases to 79.9–80.3%. On ALARM, where each increase in coverage introduces more concepts in absolute terms, the trend is more pronounced: task accuracy increases on average from 36.5–38.5% at 37.0% coverage to 47.6–49.7% at 65.8%, albeit with higher variance, and to 73.0–73.7% at full coverage. Overall, these results show that F-CMs can progressively exploit additional concept supervision as the federation expands, with larger gains when growth introduces a larger amount of new semantic information.

D.8 Robustness to imperfect concept supervision

We evaluate F-CMs under two complementary forms of imperfect concept supervision: inter-client semantic disagreement, where the same concept may have inconsistent meanings across clients (App. D.8.1), and sample-level missing or noisy concept annotations (App. D.8.2).

D.8.1 Inter-client semantic disagreement

To assess whether F-CMs remain reliable when different clients attach partially inconsistent meanings to the same concepts, we conducted an additional ablation simulating *inter-client semantic disagreement*. The goal is to model scenarios in which the same concept label does not correspond to exactly the same underlying semantics across clients, e.g., due to annotation inconsistencies, protocol differences, or distributional shifts. Concretely, for each experiment we first select a subset

Table 13: **Sensitivity to concept-space expansion.** Structural concept coverage (S.C. Cov., %) and task accuracy (T., %) as additional concepts become available during federation growth. The coverage-range column denotes the target fraction of concepts from the ground-truth graph available to the federation. Results are reported over three random seeds.

Coverage range	Method	ASIA		ALARM	
		S.C. Cov.	T.	S.C. Cov.	T.
$< 51\%$	S-F-CMs (CEM)	45.8 ± 4.2	71.3 ± 7.8	37.0 ± 3.3	36.5 ± 1.6
	S-F-CMs (C^2BM)	45.8 ± 4.2	66.7 ± 12.3	37.0 ± 3.3	38.5 ± 4.5
51–75%	F-CMs (CEM)	62.5	76.1 ± 3.1	65.8 ± 3.9	49.7 ± 12.3
	F-CMs (C^2BM)	62.5	80.2 ± 1.1	65.8 ± 3.9	47.6 ± 13.4
$> 75\%$	F-CMs (CEM)	100.0	80.3 ± 1.1	100.0	73.7 ± 0.7
	F-CMs (C^2BM)	100.0	79.9 ± 0.9	100.0	73.0 ± 0.4

of concepts and, for each selected concept, a subset of clients. Within those selected client–concept pairs, we then swap a fraction of the corresponding concept labels. In this way, the same concept may acquire conflicting label semantics across clients, while leaving the rest of the supervision unchanged. We consider two disagreement regimes: a *mild* setting, where 30% of concepts and 30% of clients are selected, and a *severe* setting, where 60% of concepts and 60% of clients are selected. For each regime, we vary the concept-swap probability in $\{30\%, 60\%, 90\%, 100\%\}$.

Figure 16 reports label accuracy for two representative F-CM instantiations, the bipartite F-CMs (CEM) and the graph-based F-CMs (C^2BM), on ASIA and SIIM-PNEUMOTHORAX. Under mild disagreement, both variants remain highly stable: performance changes only marginally even as the swap probability increases, indicating that aggregation across the remaining consistent clients is sufficient to absorb limited semantic mismatch. Under severe disagreement, performance degrades more visibly as the swap probability grows, with the decline being particularly pronounced on the smaller ASIA benchmark. On SIIM-PNEUMOTHORAX, the degradation is smoother, suggesting that larger or more redundant datasets provide some resilience even when semantic conflicts become widespread.

Overall, these results indicate that F-CMs are robust to *moderate* inter-client semantic disagreement, but that widespread and systematic concept mismatch eventually harms downstream prediction, as expected. This highlights an important practical direction for future work: detecting client-level semantic inconsistencies before aggregation. Since F-CMs operate modularly, they naturally expose module-level update patterns that could be used to identify semantic or protocol mismatches and trigger targeted diagnostics or filtering.

D.8.2 Missing and noisy concept annotations

F-CMs already operate under heterogeneous and partial concept supervision, with different clients supervising different subsets of concepts. We further assess robustness to sample-level unreliability in the available concept annotations by perturbing only the training annotations, while keeping all other experimental information unchanged. We consider two complementary settings: *missing annotations*, where each available concept annotation is independently masked with probability $\mu \in \{0.3, 0.6, 0.9\}$, and *noisy annotations*, where each available annotation is independently corrupted with probability $\eta \in \{0.3, 0.6, 0.9\}$ (e.g., by label swapping for binary concepts).

We report task and concept accuracy for CEM and C^2BM on ASIA and ALARM, averaged over three random seeds. Since these perturbations modify the sample-level annotations while leaving the client concept sets unchanged, concept coverage is unaffected by construction.

Table 14 shows substantial robustness to missing supervision. On ASIA, even masking 90% of the available concept annotations leaves task accuracy essentially unchanged for both architectures, while concept accuracy decreases by only 1.8 percentage points. On ALARM, C^2BM remains similarly robust, with task accuracy decreasing by only 1.7 points at $\mu = 0.9$, whereas CEM is more sensitive, decreasing from 69.9% to 56.8%. This suggests that structural conditioning can provide useful support when concept supervision becomes particularly sparse.

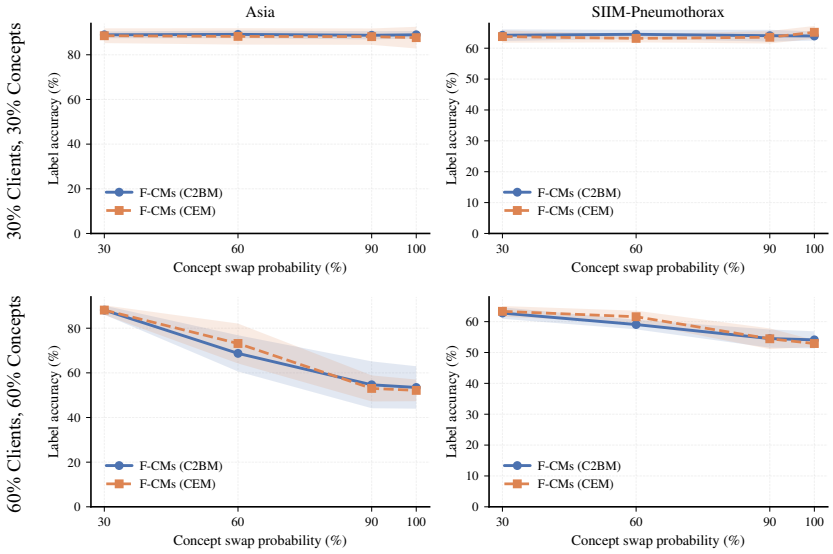


Figure 16: **Label accuracy (%) under inter-client semantic disagreement.** Results for two representative F-CM instantiations, the bipartite F-CMs (CEM) and the graph-based F-CMs (C^2BM), under increasing fractions of within-client concept-label swaps. Each row corresponds to a disagreement regime: 30% of concepts and 30% of clients selected (top), or 60% of concepts and 60% of clients selected (bottom). For each selected concept, the corresponding fraction of clients is chosen, and within them a fraction of the concept labels (x-axis: 30%, 60%, 90%, 100%) is swapped, simulating conflicting concept semantics across clients. Each column corresponds to a different dataset.

Table 14: **Robustness to missing concept annotations.** Task/concept accuracy (%, mean $\pm$ std.) under increasing probabilities μ of masking available concept annotations. Results are averaged over three random seeds.

Dataset	Model	Clean ref.	$\mu = 0.3$	$\mu = 0.6$	$\mu = 0.9$
ALARM	C^2BM	$71.8 \pm 0.9/89.9 \pm 0.5$	$72.4 \pm 0.7/89.7 \pm 0.2$	$72.6 \pm 0.3/89.1 \pm 0.1$	$70.1 \pm 1.1/85.1 \pm 0.9$
	CEM	$69.9 \pm 2.8/90.2 \pm 0.3$	$62.3 \pm 9.6/90.0 \pm 0.4$	$58.1 \pm 9.5/89.4 \pm 0.2$	$56.8 \pm 9.2/86.8 \pm 0.3$
ASIA	C^2BM	$81.0 \pm 1.6/91.2 \pm 0.6$	$81.1 \pm 1.7/91.0 \pm 0.8$	$81.1 \pm 1.9/90.7 \pm 0.6$	$81.2 \pm 1.8/89.4 \pm 0.7$
	CEM	$81.3 \pm 1.4/91.0 \pm 0.7$	$81.2 \pm 1.5/90.9 \pm 0.6$	$81.3 \pm 1.6/90.7 \pm 0.8$	$81.0 \pm 1.5/89.2 \pm 1.0$

Table 15 shows that moderate annotation noise is also well tolerated: at $\eta = 0.3$, task accuracy remains within 3.2 percentage points of the clean reference across all settings, while concept accuracy decreases by only 1.1–2.4 points. At $\eta = 0.6$, concept accuracy deteriorates substantially, although task accuracy remains close to the clean reference for both ALARM models and ASIA–CEM; ASIA–C^2BM is more sensitive in this regime. At $\eta = 0.9$, both concept and task performance degrade more consistently. Overall, noisy annotations are more harmful than missing ones, as incorrect labels introduce contradictory supervision rather than simply reducing the amount of available supervision.

D.9 Robustness to graph perturbations

In this section, we evaluate the sensitivity of the DAG aggregation procedure (Sec. 3.2.1) to noise in client-provided graphs, which can arise both from limited local data (estimation error) and from adversarial behavior (e.g., poisoning the communicated structure) [25, 71]. To reduce variance due to a specific client partition, we run this experiment with a large federation and high per-round participation. Concretely, training follows the same temporal non-stationarity setup as in the main paper, but with $|\mathcal{K}_t| = 100$ participating clients per round. The client pool is time-varying: during a warm-up phase the federation contains $|\mathcal{K}(t)| = 100$ clients, and at the predefined join round (as in our standard protocol, depending on the dataset) an additional cohort joins, expanding the pool to

Table 15: **Robustness to noisy concept annotations.** Task/concept accuracy (%, mean ± std.) under increasing probabilities η of corrupting available concept annotations. Results are averaged over three random seeds.

Dataset	Model	Clean ref.	$\eta = 0.3$	$\eta = 0.6$	$\eta = 0.9$
ALARM	C^2BM	$71.4 \pm 0.9/89.9 \pm 0.5$	$71.7 \pm 0.5/87.9 \pm 0.4$	$72.1 \pm 0.9/46.7 \pm 2.4$	$51.7 \pm 9.6/14.9 \pm 2.2$
	CEM	$69.9 \pm 2.8/90.2 \pm 0.3$	$73.1 \pm 0.6/89.1 \pm 0.6$	$71.6 \pm 1.3/47.1 \pm 2.7$	$50.5 \pm 12.2/7.2 \pm 0.7$
ASIA	C^2BM	$81.0 \pm 1.6/91.2 \pm 0.6$	$81.3 \pm 1.3/89.3 \pm 1.0$	$64.1 \pm 8.4/30.6 \pm 9.4$	$63.1 \pm 7.4/24.9 \pm 5.9$
	CEM	$81.3 \pm 1.4/91.0 \pm 0.7$	$81.7 \pm 1.4/88.6 \pm 1.0$	$81.3 \pm 1.2/16.1 \pm 1.8$	$72.5 \pm 7.2/15.5 \pm 1.8$

$|\mathcal{K}(t)| = 200$. Late-arriving clients may introduce new concepts and associated edges, requiring the aggregated shared structure to expand accordingly.

Each participating client provides a local graph (binary adjacency matrix) over its observed variables. We inject noise by randomly deleting, adding, or reversing edges, while enforcing acyclicity. We vary two factors: *(% client alteration)*, the fraction of clients whose graphs are corrupted, and *(graph alteration)*, the fraction of edges modified within each corrupted client graph. Aggregation quality is measured by *DiffPairs*, i.e., the number of node pairs whose relative ordering differs between the aggregated shared DAG and the reference DAG (lower is better). Figure 17 reports results for the federated regimes (F-CMs and S-F-CMs) and includes localized training (Loc.) as a reference.

Empirical Validation (Graph Quality). Overall, the proposed server-side graph aggregation is robust to substantial client-side noise. In particular, for moderate perturbations (e.g., 30% graph alteration), the aggregated structure remains essentially unchanged across the full range of % client alteration: DiffPairs stays nearly constant for both F-CMs and S-F-CMs. This indicates that the aggregation effectively denoises inconsistent local proposals when the per-client corruption level is limited. In contrast, the Localized baseline exhibits a markedly different behavior: for a fixed graph alteration level, DiffPairs increases approximately linearly with % client alteration, reflecting the lack of cross-client averaging and the direct propagation of local graph errors to the deployed structure.

At higher corruption levels, DiffPairs increases as expected. For 60% graph alteration, degradation becomes visible once a large majority of clients are corrupted (typically around 60–70% corrupted clients across datasets). For 90% graph alteration, the effect appears earlier, becoming pronounced already around 50% corrupted clients. Finally, the gap between F-CMs and S-F-CMs is primarily explained by *structure evolution*: S-F-CMs does not update the shared graph when late-arriving clients introduce new concepts and edges, leading to an increasingly partial global view, whereas F-CMs continuously integrates new proposals and maintains coverage of the evolving federation.

Empirical Validation (Downstream Accuracy). To directly assess whether structural errors in the aggregated graph translate into degraded downstream performance, we ran an additional experiment under the same perturbation protocol and measured the final task accuracy of C^2BM, used here as a representative graph-based instantiation. Table 16 reports results for the setting with 60% graph alteration as the fraction of corrupted clients increases.

The results show that downstream performance is substantially more robust than graph quality alone might suggest. On SIIM-PNEUMOTHORAX, F-CMs changes only from 68.80 ± 2.51 at 0% corrupted clients to 67.06 ± 2.90 at 90% corrupted clients, despite the visible degradation in DiffPairs at high corruption levels. On ASIA, task accuracy remains essentially unchanged around 80% across the full corruption range. S-F-CMs remain roughly stable, but at a substantially lower absolute accuracy level, consistent with the static-structure limitation discussed above. Overall, these results indicate that increases in DiffPairs do not imply comparable drops in downstream task accuracy: F-CMs remain robust to moderate structural misspecification, and graph degradation becomes noticeable earlier than any meaningful loss in predictive performance.

D.10 Ablation of graph aggregation strategies under graph perturbations

To complement the robustness analysis in App. D.9, we further compare different server-side graph aggregation strategies under the same noisy FL setting. In particular, we contrast our proposed aggregation with standard alternatives based on UNION [72], INTERSECTION [72], and BAYESIAN

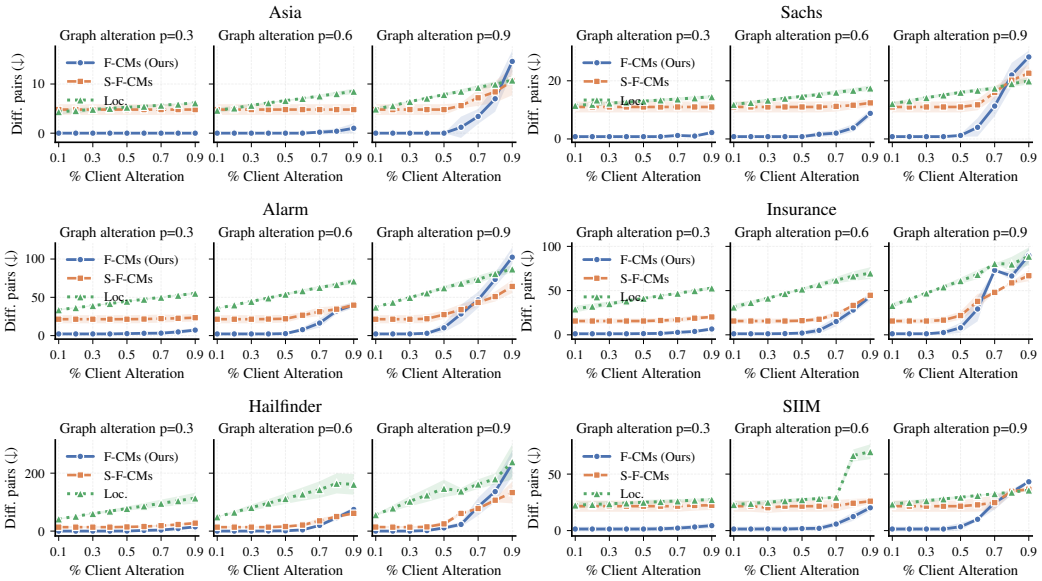


Figure 17: **Robustness of DAG aggregation to client graph perturbations.** DiffPairs (lower is better) between the aggregated DAG and the ground-truth DAG as a function of % client alteration and per-client graph alteration.

Table 16: **Downstream robustness to graph misspecification.** Task accuracy (%, mean ± std.) for C^{2}BM under the same perturbation protocol as Figure 17, with *graph alteration* fixed at 60% and varying fractions of corrupted clients. Although graph quality degrades at higher corruption levels, downstream task accuracy remains largely stable, especially for F-CMs.

Dataset	Method	0%	30%	60%	90%
ASIA	F-CMs (C^{2}BM)	80.35±2.16	80.41±1.80	80.25±1.98	80.39±2.51
	S-F-CMs (C^{2}BM)	59.73±20.19	59.99±20.15	59.99±20.15	60.83±21.46
SIIM-PNEUMOTHORAX	F-CMs (C^{2}BM)	68.80±2.51	68.03±3.34	68.62±2.62	67.06±2.90
	S-F-CMs (C^{2}BM)	62.68±12.67	62.13±1.01	62.14±1.02	63.07±2.58

NETWORK STRUCTURE FUSION (BN fusion) [73], and evaluate each of them under three levels of structural visibility: *localized* training (LOC.), *static federated* visibility (S-*), and *dynamic federated* visibility, where the shared graph is continuously updated as new clients join and introduce additional concepts and dependencies. As in App. D.9, all experiments follow the same temporal non-stationarity protocol, with large federation size and high participation per round, and performance is measured with *DiffPairs* (lower is better). We additionally assess whether reliability-aware client weighting can further improve robustness within our aggregation rule.

Figures 18 and 19 report the results for two perturbation regimes, corresponding to moderate corruption (*graph alteration* $p = 0.3$, *% client alteration* $= 0.3$) and stronger corruption (*graph alteration* $p = 0.6$, *% client alteration* $= 0.6$). The color coding highlights the visibility regime: blue for dynamic training, orange for static federated visibility, and green for localized visibility.

Comparison of aggregation strategies. The results highlight three main trends. First, our aggregation is consistently the most robust across both perturbation regimes. In all datasets and in both Figures 18 and 19, F-CMS aggregation achieves the lowest DiffPairs, while S-F-CMS is consistently the strongest static alternative. Moreover, S-F-CMS always improves over LOC., and the dynamic version further improves over the static one on all datasets, confirming that our aggregation can exploit the broader structural visibility provided by federation and by late-arriving clients.

Second, broader visibility is *not* uniformly beneficial across aggregation rules. While moving from local to static federated visibility can help some baselines, the effect is not systematic, and in several cases local training remains preferable to federated aggregation with naive fusion rules. This is

particularly visible for UNION, and to a lesser extent for INTERSECTION. In other words, access to more client-side graphs does not automatically improve the recovered shared structure: when the aggregation rule is poorly matched to the underlying heterogeneity, the additional structural proposals may introduce more inconsistencies than useful signal. This effect becomes even clearer when comparing static and dynamic visibility. For UNION and BN FUSION, dynamic visibility often increases DiffPairs, in some cases substantially, especially under stronger corruption (Figure 19). A plausible explanation is that these rules do not sufficiently denoise discordant client proposals. By contrast, our aggregation operates edge-wise, considers only clients that observe both nodes, and selects among $\{\mathcal{V}_i \to \mathcal{V}_j,\ \mathcal{V}_j \to \mathcal{V}_i,\ \varnothing\}$, allowing the server to reject weak or inconsistent evidence instead of forcing unreliable edges into the aggregate. As more clients are observed, including corrupted ones and late-arriving clients with partially different structural patterns, UNION-style aggregation becomes overly permissive, accumulating spurious edges and incorrect orderings, while BN FUSION appears similarly sensitive to conflicting local structures. INTERSECTION, by contrast, is more conservative and therefore more stable, but its gains from dynamic visibility are limited.

Finally, these results clarify that the benefit of global visibility depends on the aggregation mechanism. More visibility is useful only when the fusion rule can transform heterogeneous local proposals into a coherent shared DAG. Our method does so reliably: unlike alternative graph fusion strategies, it benefits monotonically from richer structural visibility and degrades more gracefully as both the fraction of corrupted clients and the per-client graph alteration increase. Overall, the ablation reinforces the central claim of this work: in evolving federated concept-based learning, global structural visibility is valuable only when coupled with a robust, structure-aware aggregation rule.

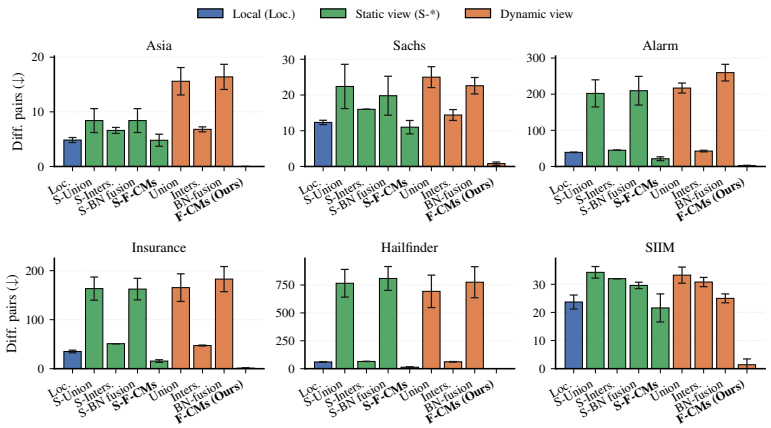


Figure 18: **Ablation of graph aggregation strategies under moderate perturbations.** DiffPairs (lower is better) for different aggregation rules under localized, static federated, and dynamic federated visibility, with graph alteration $p = 0.3$ and 30% corrupted clients.

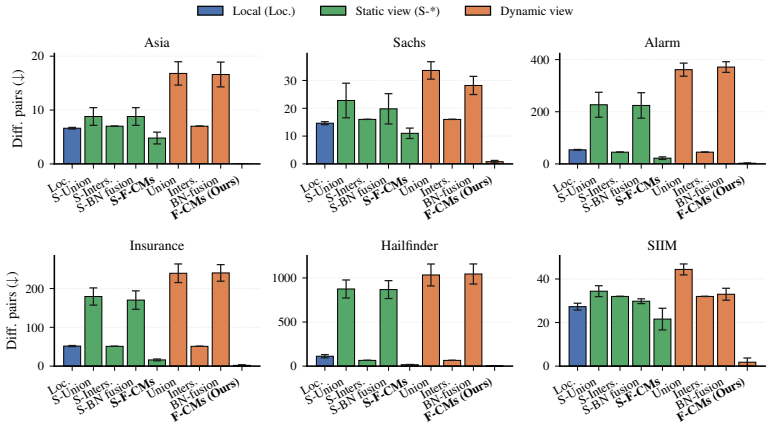


Figure 19: **Ablation of graph aggregation strategies under stronger perturbations.** DiffPairs (lower is better) for different aggregation rules under localized, static federated, and dynamic federated visibility, with graph alteration $p = 0.6$ and 60% corrupted clients.

Table 17: **Consensus-based client weighting under graph perturbations.** DiffPairs (lower is better) for consensus-based and data-size-based graph aggregation under varying client alteration (CA) and graph alteration (GA).

		Consensus-based aggregation			Data-size-based aggregation		
Dataset	CA	GA= 0.3	GA= 0.6	GA= 0.9	GA= 0.3	GA= 0.6	GA= 0.9
ASIA	0.3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	0.6	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 0.6	0.0 ± 0.0	0.0 ± 0.0	1.0 ± 1.7
	0.9	0.0 ± 0.0	0.3 ± 0.6	14.3 ± 1.2	0.0 ± 0.0	1.0 ± 1.0	14.3 ± 1.2
ALARM	0.3	1.7 ± 1.5	1.7 ± 1.5	1.7 ± 1.5	1.7 ± 1.5	1.7 ± 1.5	1.7 ± 1.5
	0.6	2.0 ± 1.0	2.7 ± 0.6	7.0 ± 2.7	2.3 ± 1.5	7.7 ± 2.3	26.7 ± 13.1
	0.9	6.3 ± 1.5	39.7 ± 3.5	90.0 ± 16.5	7.0 ± 1.7	41.7 ± 2.1	93.7 ± 11.0

Reliability-aware client weighting. Our aggregation rule supports generic client weights α_k (Sec. 3.2.1). In our experiments, following the standard FedAvg weighting scheme, we use data-size-based weighting, i.e., clients are weighted proportionally to their local dataset sizes. More generally, F-CMs can incorporate alternative weighting strategies. We therefore design a consensus-based weighting strategy that estimates client reliability from the agreement among their reported structures.

For each client k and informative variable pair $\{V_i, V_j\}$ observed by the client, we define the pairwise agreement score $a_{k,ij}$ as the fraction of other clients observing the same pair that report the same relation as client k. We consider a pair informative when it is observed by at least two clients and at least one of them predicts an edge, thereby preventing absent edges from dominating the agreement score. The consensus weight of client k is then defined as

$$\alpha_k^{\mathrm{CB}} = \frac{1}{|\mathcal{I}_k|} \sum_{\{V_i,V_j\} \in \mathcal{I}_k} a_{k,ij},$$

where $\mathcal{I}_k$ denotes the set of informative pairs available to client k. Clients whose structural predictions systematically agree with the federation therefore receive greater weight during graph aggregation.

Table 17 shows that both weighting strategies are highly robust under mild structural perturbations, while consensus-based weighting (CB) provides additional resilience as corruption increases. On ASIA, the two strategies are nearly indistinguishable, with CB reducing DiffPairs from 1.0 to 0.3 only at CA$= 0.9$, GA$= 0.6$. The difference is more pronounced on ALARM: at CA$= 0.6$, GA$= 0.9$, CB reduces DiffPairs from 26.7 to 7.0, while smaller improvements are observed across most other corrupted settings. Under the most extreme regime (CA$= 0.9$, GA$= 0.9$), both methods eventually degrade substantially. Overall, consensus-based weighting performs comparably to data-size weighting in low-noise regimes and can improve robustness when a larger fraction of client structures is unreliable, by reducing the influence of systematically discordant reports.

D.11 Multimodal extension

We extend F-CMs to multimodal federated settings, where different clients may observe different input modalities while the semantic prediction targets, i.e., concepts and task labels, are shared across modalities. Fig. 20 provides an overview.

To experimentally validate this extension, we consider CHEXPERT PLUS (App. A.4), which augments CHEXPERT (App. A.3) with radiology reports. In what follows, we specify the modified problem setting, architecture, local optimization, aggregation and experimental setup.

Multimodal problem setting. Let $\mathcal{A}$ denote the set of available modalities. In CHEXPERT PLUS, we consider the image modality I (chest x-ray images) and the text modality T (radiology reports). Each client k observes data from a single modality $a^{(k)} \in \mathcal{A}$ and holds a local dataset

$$D^{(k)} = \left\{ \left(x_i^{(k,a^{(k)})}, \mathbf{v}_i^{(k)} \right) \right\}_{i=1}^{n^{(k)}},$$

where $x_i^{(k,a^{(k)})} \in \mathcal{X}^{a^{(k)}}$ is the input observed under modality $a^{(k)}$, and $\mathbf{v}_i^{(k)}$ contains the supervision available for example i. As in the unimodal case, each client supervises a subset of interpretable variables $\mathcal{V}^{(k)}$, which may include concepts, task variable, or both. Importantly, concepts and task

labels are treated as modality-independent interpretable targets. Each concept C_j and the task Y are assumed to be predictable, in principle, from any modality $a \in \mathcal{A}$. Thus, the same clinical concept, such as *Cardiomegaly*, can be predicted from a chest X-ray image or from a radiology report.

Multimodal F-CM shared model. The main architectural change is that the single input encoder is replaced by a collection of modality-specific encoders. For each modality $a \in \mathcal{A}$, we define an encoder

$$g^a_{\theta^a_t} : \mathcal{X}^a \to \mathbb{R}^{d_a},$$

which maps inputs from modality a to modality-specific embeddings. Since different modalities produce embeddings with different statistical and geometric properties, we introduce a modality-alignment module

$$q^a_{\eta^a_t} : \mathbb{R}^{d_a} \to \mathbb{R}^d,$$

which maps modality-specific embeddings into a shared latent space. For an example observed by client k, the shared representation is therefore

$$z_i^{(k)} = q^{a^{(k)}}_{\eta^{a^{(k)}}_t} \left(g^{a^{(k)}}_{\theta^{a^{(k)}}_t} \left(x_i^{(k,a^{(k)})} \right) \right) \in \mathbb{R}^d.$$

where $a^{(k)}$ denotes the modality observed by client k. Thus, $g^{a^{(k)}}_{\theta^{a^{(k)}}_t}$ and $q^{a^{(k)}}_{\eta^{a^{(k)}}_t}$ are respectively the encoder and alignment module associated with that modality. In our implementation, the image encoder and the text encoder are instantiated as reported in App. A.4, while each modality-specific alignment module is parameterized as an MLP.

The concept and task modules are the same as in the unimodal case and are shared across modalities. Therefore, at round t, the multimodal F-CM is characterized by the parameters

$$\mathbf{w}'_t = \left(\{\theta^a_t\}_{a \in \mathcal{A}}, \{\eta^a_t\}_{a \in \mathcal{A}}, \phi_t, \psi_t \right).$$

Graph aggregation and architecture adaptation. The semantic graph remains modality-invariant. Client graphs encode dependencies among semantic variables, not among raw input modalities. Therefore, graph aggregation proceeds exactly as in the unimodal case. Dynamic architecture adaptation is also preserved. The only additional mechanism introduced by the multimodal extension is expansion along the modality axis: if a new modality appears at round t, the server initializes a new modality-specific encoder $g^a_{\theta^a_t}$ and alignment module $q^a_{\eta^a_t}$, while reusing the existing semantic modules whenever possible.

Client-side optimization. At each round t, the server broadcasts the current global model to the selected clients. Each client k processes its local inputs through the modality-specific encoder and alignment module associated with its modality $a^{(k)}$. Client-side optimization then follows the same module-specific principle as in the unimodal setting: the client updates only the trainable modules for which it provides supervision. Concept and task modules without local supervision remain frozen.

Server-side aggregation. Aggregation is performed module-wise. For each modality a, the server aggregates the encoder and alignment module only over clients that observe modality a, using the same weighted averaging rule as in the unimodal setting. If no selected client observes modality a at round t, the corresponding alignment module is left unchanged.

Concept and task modules are aggregated as in unimodal F-CMs. In particular, each concept module is aggregated only over clients that provide supervision for the corresponding concept, while the task module is aggregated only over clients that provide task supervision.

Notably, this section is intended as a first proof-of-concept extension of F-CMs to multimodal federated settings. Further dedicated multimodal alignment objectives could be explored, for instance by adding an explicit regularization term that encourages the latent representations of different modalities to be closer in the shared embedding space.

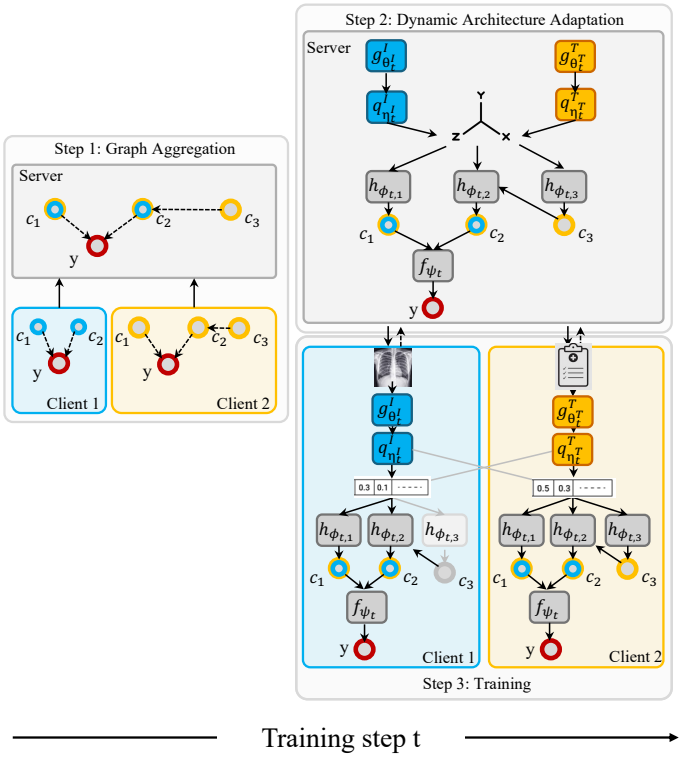


Figure 20: **Multimodal F-CMs overview**. At each round t, multimodal F-CMs follow the same steps as standard F-CMs: graph aggregation, dynamic architecture adaptation, and training. To handle heterogeneous input modalities, the shared CM includes, for each modality, a modality-specific encoder and a modality-alignment module, shown as the blue and yellow modules in the figure. The modality-specific encoder produces modality-specific embeddings, while the modality-alignment module maps them into a common latent space. Each client then processes its local inputs using only the encoder and alignment module associated with its modality, obtaining a latent representation that is then used to predict concepts and the task label.

Experimental evaluation. We empirically evaluate the multimodal extension on CHEXPERT PLUS by measuring task accuracy and concept coverage under the same training regimes and experimental setup used in the unimodal evaluation. Table 18 reports results for the most relevant baselines, namely the centralized, S-F-CMs, and F-CMs instantiations.

The results follow the same trend observed in the unimodal setting. S-F-CMs aggregate client updates but keep the semantic architecture fixed, resulting in limited coverage when later clients introduce additional concept supervision. In contrast, F-CMs dynamically adapt the model structure and recover full coverage of task-relevant concepts across all architectures.

Table 18: **Task accuracy** (%) and **concept coverage** (%) with respect to the ground-truth concepts causally relevant for the task.

	Cent.		**S-F-CMs**		**F-CMs**	
	Task Acc.	**Concept Cov.**	**Task Acc.**	**Concept Cov.**	**Task Acc.**	**Concept Cov.**
OpaqNN	72.9 ± 0.5	100	58.4 ± 2.6	52.0 ± 10.2	70.3 ± 2.8	100
C^2BM	73.3 ± 0.6	100	58.4 ± 2.5	60.0 ± 8.2	66.1 ± 2.7	100
CBM	71.3 ± 0.1	100	55.4 ± 3.0	52.0 ± 10.2	63.3 ± 1.4	100
CEM	73.2 ± 0.3	100	55.7 ± 3.1	52.0 ± 10.2	64.8 ± 2.0	100
CGM	73.3 ± 0.4	100	60.0 ± 2.9	60.0 ± 8.2	62.3 ± 1.9	100

E Algorithmic details and analysis

E.1 Algorithm

Algorithm 1 summarizes the end-to-end F-CMs training loop under statistical heterogeneity and temporal non-stationarity. At each round, the server samples available clients, aggregates their supervised variables (and optional local structure) into an updated shared variable set and DAG, and then adapts the shared CM architecture modularly. Clients perform module-specific local updates by freezing unsupervised components, and the server aggregates updates module-wise, averaging each module only over clients that contributed to it.

Algorithm 1: F-CMs: Federated Concept-based Models.

Input: Initial shared parameters $\mathbf{w}_0 = (\theta_0, \phi_0, \psi_0)$, variable set $\mathcal{V}_0$, shared DAG $\mathcal{G}_0$, number of rounds T
Output: Final shared model parameters $\mathbf{w}_T$ and shared structure $(\mathcal{V}_T, \mathcal{G}_T)$

Initialize cache $\mathcal{H}_0$ with the latest available client graphs/adjacency matrices;
for $t = 0, 1, \ldots, T-1$ **do**
  $\mathcal{K}_t \leftarrow$ ClientSampling($\mathcal{K}(t)$);
  `// Client-side: local graph proposals`
  for ***each client*** $k \in \mathcal{K}_t$ ***in parallel*** **do**
    if *client k is new or has updated its local structure* **then**
      Send weighted adjacency matrix $A^{(k)}$ to server;
    Send supervised variables $\mathcal{V}^{(k)}$ to server;
  `// Server-side: shared structure update`
  Update cache $\mathcal{H}_{t+1}$ with newly received $A^{(k)}$;
  $\mathcal{V}_{t+1} \leftarrow \mathcal{V}_t \cup \bigcup_{k \in \mathcal{K}_t} \mathcal{V}^{(k)}$;
  $\mathcal{G}_{t+1} \leftarrow$ AggregateDAG$\big(\mathcal{H}_{t+1}, \mathcal{V}_{t+1}, \{\alpha_k\}\big)$;
  $\tilde{\mathbf{w}}_t \leftarrow$ AdaptModel$(\mathbf{w}_t, \mathcal{V}_{t+1}, \mathcal{G}_{t+1})$;
  Broadcast $(\tilde{\mathbf{w}}_t, \mathcal{G}_{t+1}, \mathcal{V}_{t+1})$ to all clients in $\mathcal{K}_t$;
  `// Client-side: module-specific training`
  for ***each client*** $k \in \mathcal{K}_t$ ***in parallel*** **do**
    $\tilde{\mathbf{w}}_t^{(k)} \leftarrow$ FreezeUnsup$\big(\tilde{\mathbf{w}}_t, \mathcal{G}_{t+1}, \mathcal{V}^{(k)}\big)$;
    $\Delta\tilde{\mathbf{w}}_t^{(k)} \leftarrow$ LocalUpdate$\big(\tilde{\mathbf{w}}_t^{(k)}, \mathcal{G}_{t+1}, \mathcal{V}^{(k)}, \mathcal{D}^{(k)}\big)$;
    Send $\Delta\tilde{\mathbf{w}}_t^{(k)}$ to server;
  `// Server-side: module-wise aggregation`
  $\mathbf{w}_{t+1} \leftarrow$ ModuleWiseFedAvg$\big(\tilde{\mathbf{w}}_t, \{\Delta\tilde{\mathbf{w}}_t^{(k)}\}_{k \in \mathcal{K}_t}, \{\mathcal{V}^{(k)}\}_{k \in \mathcal{K}_t}\big)$;

E.2 Theoretical analysis

We consider a static setting with a fixed set of clients $\mathcal{K}$ and fixed supervision patterns, i.e., the variables supervised by each client remain fixed throughout training. The model architecture is therefore fixed and contains one module for each variable supervised by at least one client in $\mathcal{K}$. Each client $k \in \mathcal{K}$ has a local objective $F^{(k)}$.

Partial supervision. Since clients may supervise only a subset of variables, they update only the block of parameters associated with the modules that predict the corresponding variables. We encode these locally updated coordinates through a diagonal mask $P^{(k)}$, whose entries are one for parameters updated by client k and zero otherwise. For example, if the model contains modules for predicting (C_1, C_2, Y) and client k has supervision only for C_1 and Y, then $P^{(k)}$ selects the parameters of the modules for C_1 and Y and masks out the parameters of the C_2-module. In our analysis, these supervision patterns, and hence the masks $P^{(k)}$, remain fixed throughout training.

Module-wise server aggregation. The server aggregates parameters module-wise. For a module j, let $\mathcal{K}^j \subseteq \mathcal{K}$ be the clients that update that module. The server averages the parameters of module j

only across clients in $\mathcal{K}^j$, using weights

$$\beta_j^{(k)} = \frac{n^{(k)}}{\sum_{r \in \mathcal{K}^j} n^{(r)}}, \qquad k \in \mathcal{K}^j.$$

To model this, we first define for each client k, a diagonal module-wise aggregation operator $R^{(k)}$ that stores the aggregation weights for client k: on the coordinates of module j, its diagonal entries are $\beta_j^{(k)}$ if client k updates module j, and zero otherwise. Continuing the example above, if client k supervises only C_1 and Y, then $R^{(k)}$ places the weights $\beta_{C_1}^{(k)}$ and $\beta_Y^{(k)}$ on the corresponding parameter blocks, i.e., parameters of the modules for predicting C_1 and Y, and zero weight on the C_2-parameter block. Thus, $P^{(k)}$ tells which modules client k updates locally, while $R^{(k)}$ tells how much those local updates count in the server aggregation.

Local optimization. For each client k, let

$$g^{(k)}(\mathbf{w}; \xi^{(k)})$$

denote a stochastic gradient estimator of the local objective $F^{(k)}$ at parameter vector $\mathbf{w}$, computed on a random sample or minibatch $\xi^{(k)}$.

At communication round t, the server broadcasts $\mathbf{w}_t$ and each client initializes

$$\mathbf{w}_{t,0}^{(k)} = \mathbf{w}_t.$$

Each client then performs E local stochastic-gradient steps. At local step e, client k draws a sample or minibatch $\xi_{t,e}^{(k)}$ and computes

$$g_{t,e}^{(k)} := g^{(k)}(\mathbf{w}_{t,e}^{(k)}; \xi_{t,e}^{(k)}).$$

The masked local update is

$$\mathbf{w}_{t,e+1}^{(k)} = \mathbf{w}_{t,e}^{(k)} - \eta P^{(k)} g_{t,e}^{(k)}, \qquad e = 0, \dots, E-1.$$

The server aggregates the final local parameters module-wise. To write this update compactly, we unroll the local steps. For any module j updated by client k, we have

$$\mathbf{w}_{t,E}^{(k),j} = \mathbf{w}_t^j - \eta \sum_{e=0}^{E-1} g_{t,e}^{(k),j}.$$

Therefore, the module-wise aggregation rule gives

$$\mathbf{w}_{t+1}^j = \sum_{k \in \mathcal{K}^j} \beta_j^{(k)} \mathbf{w}_{t,E}^{(k),j} = \mathbf{w}_t^j - \eta \sum_{e=0}^{E-1} \sum_{k \in \mathcal{K}^j} \beta_j^{(k)} g_{t,e}^{(k),j}.$$

In vector form, this can be written as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \sum_{e=0}^{E-1} \sum_{k \in \mathcal{K}} R^{(k)} g_{t,e}^{(k)}.$$

Assumptions. We make the following assumptions.

(S1) **L-smoothness** Each local objective $F^{(k)}$ is differentiable and L-smooth, i.e.,

$$\|\nabla F^{(k)}(\mathbf{u}) - \nabla F^{(k)}(\mathbf{v})\| \le L\|\mathbf{u} - \mathbf{v}\|, \qquad \forall \mathbf{u}, \mathbf{v},\ \forall k \in \mathcal{K}.$$

We also assume that there exists a differentiable, L-smooth, and lower-bounded effective global objective F, with $F(w) \ge F_{\inf}$.

(S2) **Unbiased stochastic gradients with bounded variance.** For each client k, the stochastic gradient estimator is unbiased on the locally updated coordinates:

$$\mathbb{E}_{\xi^{(k)}}\left[P^{(k)} g^{(k)}(\mathbf{w}; \xi^{(k)})\right] = P^{(k)} \nabla F^{(k)}(\mathbf{w}).$$

Moreover, its variance on those coordinates is uniformly bounded: there exists $\sigma^2 \geq 0$ such that

$$\mathbb{E}_{\xi^{(k)}}\left[\left\|P^{(k)}\big(g^{(k)}(\mathbf{w};\xi^{(k)}) - \nabla F^{(k)}(\mathbf{w})\big)\right\|^2\right] \leq \sigma^2$$

for every client k and every $\mathbf{w}$. Conditioned on the current local iterates, the stochastic gradients are independent across clients and local steps.

(S3) **Bounded module-wise alignment mismatch.** We assume that the module-wise aggregated federation direction is close to the gradient of the global objective. Specifically, there exists $\zeta^2 \geq 0$ such that, for every $\mathbf{w}$,

$$\left\|\sum_{k\in\mathcal{K}} R^{(k)}\nabla F^{(k)}(\mathbf{w}) - \nabla F(\mathbf{w})\right\|^2 \leq \zeta^2.$$

The case $\zeta = 0$ corresponds to exact alignment between the module-wise aggregate direction and the gradient of F.

(S4) **Bounded second moment of masked stochastic gradients.** There exists $G^2 \geq 0$ such that

$$\mathbb{E}\Big[\|P^{(k)}g^{(k)}(\mathbf{w};\xi^{(k)})\|^2\Big] \leq G^2$$

for every client k and every $\mathbf{w}$. This assumption controls the drift caused by multiple local steps. Since $R^{(k)}$ is supported on the same coordinates as $P^{(k)}$ and has entries bounded by one, it also controls the corresponding module-wise aggregated updates.

Notice moreover that, by construction, for every module j, there is at least one client that supervises it, i.e., $\mathcal{K}^j \neq \emptyset$, and the aggregation weights satisfy $\sum_{k\in\mathcal{K}^j} \beta_j^{(k)} = 1$.

We next bound the discrepancy between the actual aggregated local-update direction and the ideal global direction.

Lemma E.1 (Deviation of the aggregated local direction)**.** *Define*

$$\bar{g}_t \coloneqq \frac{1}{E}\sum_{e=0}^{E-1}\sum_{k\in\mathcal{K}} R^{(k)} g_{t,e}^{(k)}$$

and

$$\delta_t \coloneqq \bar{g}_t - \nabla F(\mathbf{w}_t).$$

Under Assumptions (S1), (S2), (S3), and (S4), for every round t,

$$\mathbb{E}\|\delta_t\|^2 \leq C\left(\frac{\sigma^2}{E} + L^2\eta^2(E-1)^2G^2 + \zeta^2\right),$$

where $C > 0$ depends only on the fixed client set.

Proof. We decompose δ_t into three terms:

$$\delta_t = A_t + B_t + C_t,$$

where

$$A_t \coloneqq \frac{1}{E}\sum_{e=0}^{E-1}\sum_{k\in\mathcal{K}} R^{(k)}\big(g_{t,e}^{(k)} - \nabla F^{(k)}(\mathbf{w}_{t,e}^{(k)})\big),$$

$$B_t \coloneqq \frac{1}{E}\sum_{e=0}^{E-1}\sum_{k\in\mathcal{K}} R^{(k)}\big(\nabla F^{(k)}(\mathbf{w}_{t,e}^{(k)}) - \nabla F^{(k)}(\mathbf{w}_t)\big),$$

$$C_t \coloneqq \sum_{k\in\mathcal{K}} R^{(k)}\nabla F^{(k)}(\mathbf{w}_t) - \nabla F(\mathbf{w}_t).$$

Applying Jensen's inequality, we have

$$\mathbb{E}\big[\|\delta_t\|^2\big] \leq 3\mathbb{E}\big[\|A_t\|^2\big] + 3\mathbb{E}\big[\|B_t\|^2\big] + 3\mathbb{E}\big[\|C_t\|^2\big].$$

We bound the three terms separately.

Bound on A_t. Let us define

$$\varepsilon_{t,e}^{(k)} \coloneqq g_{t,e}^{(k)} - \nabla F^{(k)}(\mathbf{w}_{t,e}^{(k)}).$$

For each round t and local step e, let $\mathcal{F}_{t,e}$ denote the sigma-field containing all information available before the minibatches at step e are sampled. In particular, given $\mathcal{F}_{t,e}$, the local iterate $\mathbf{w}_{t,e}^{(k)}$ is fixed, since it has been computed from the minibatches used in the previous local steps. Thus, for client k, the fresh randomness in $g_{t,e}^{(k)}$ comes from the minibatch $\xi_{t,e}^{(k)}$. Therefore, by Assumption (S2), we have

$$\mathbb{E}\Big[P^{(k)} g_{t,e}^{(k)} \mid \mathcal{F}_{t,e}\Big] = P^{(k)} \nabla F^{(k)}(\mathbf{w}_{t,e}^{(k)}).$$

Equivalently,

$$\mathbb{E}\Big[P^{(k)} \varepsilon_{t,e}^{(k)} \mid \mathcal{F}_{t,e}\Big] = 0.$$

Since $R^{(k)} P^{(k)} = R^{(k)}$ and $R^{(k)}$ is fixed , this also implies

$$\mathbb{E}\Big[R^{(k)} \varepsilon_{t,e}^{(k)} \mid \mathcal{F}_{t,e}\Big] = R^{(k)} \mathbb{E}\Big[P^{(k)} \varepsilon_{t,e}^{(k)} \mid \mathcal{F}_{t,e}\Big] = 0. \tag{4}$$

Moreover, by Assumption (S2),

$$\mathbb{E}\Big[\|P^{(k)} \varepsilon_{t,e}^{(k)}\|^2 \mid \mathcal{F}_{t,e}\Big] \le \sigma^2.$$

For convenience, define

$$M_{t,e} \coloneqq \sum_{k \in \mathcal{K}} R^{(k)} \varepsilon_{t,e}^{(k)}.$$

Then

$$A_t = \frac{1}{E} \sum_{e=0}^{E-1} M_{t,e}.$$

Hence

$$\|A_t\|^2 = \frac{1}{E^2} \left\| \sum_{e=0}^{E-1} M_{t,e} \right\|^2 = \frac{1}{E^2} \sum_{e=0}^{E-1} \|M_{t,e}\|^2 + \frac{2}{E^2} \sum_{0 \le e < e' \le E-1} \langle M_{t,e}, M_{t,e'} \rangle.$$

Taking expectations gives

$$\mathbb{E}[\|A_t\|^2] = \frac{1}{E^2} \sum_{e=0}^{E-1} \mathbb{E}[\|M_{t,e}\|^2] + \frac{2}{E^2} \sum_{0 \le e < e' \le E-1} \mathbb{E}[\langle M_{t,e}, M_{t,e'} \rangle].$$

We now show that all cross-step terms vanish. Fix $e < e'$. The quantity $M_{t,e}$ is already determined before the minibatches at step e' are sampled, while, by Eq. 4,

$$\mathbb{E}[M_{t,e'} \mid \mathcal{F}_{t,e'}] = 0.$$

Given the properties of conditional expectation, we have

$$\begin{aligned} \mathbb{E}[\langle M_{t,e}, M_{t,e'} \rangle] &= \mathbb{E}[\mathbb{E}[\langle M_{t,e}, M_{t,e'} \rangle \mid \mathcal{F}_{t,e'}]] \\ &= \mathbb{E}[\langle M_{t,e}, \mathbb{E}[M_{t,e'} \mid \mathcal{F}_{t,e'}] \rangle] \\ &= 0. \end{aligned}$$

Therefore,

$$\mathbb{E}[\|A_t\|^2] = \frac{1}{E^2} \sum_{e=0}^{E-1} \mathbb{E}[\|M_{t,e}\|^2].$$

We next expand the square within a fixed local step e:

$$\|M_{t,e}\|^2 = \left\| \sum_{k \in \mathcal{K}} R^{(k)} \varepsilon_{t,e}^{(k)} \right\|^2 = \sum_{k \in \mathcal{K}} \|R^{(k)} \varepsilon_{t,e}^{(k)}\|^2 + \sum_{k \ne r} \left\langle R^{(k)} \varepsilon_{t,e}^{(k)}, R^{(r)} \varepsilon_{t,e}^{(r)} \right\rangle.$$

Taking conditional expectation with respect to $\mathcal{F}_{t,e}$, the cross-client terms vanish. Indeed, for $k \neq r$, let

$$X_k := R^{(k)}\varepsilon_{t,e}^{(k)}, \qquad X_r := R^{(r)}\varepsilon_{t,e}^{(r)}.$$

Conditioned on $\mathcal{F}_{t,e}$, the minibatch noises of different clients are independent. Moreover, as shown above,

$$\mathbb{E}[X_k \mid \mathcal{F}_{t,e}] = \mathbb{E}[X_r \mid \mathcal{F}_{t,e}] = 0.$$

Therefore,

$$\begin{aligned}\mathbb{E}\Big[\Big\langle R^{(k)}\varepsilon_{t,e}^{(k)}, R^{(r)}\varepsilon_{t,e}^{(r)}\Big\rangle \mid \mathcal{F}_{t,e}\Big] &= \langle \mathbb{E}[X_k \mid \mathcal{F}_{t,e}], \mathbb{E}[X_r \mid \mathcal{F}_{t,e}]\rangle \\ &= 0.\end{aligned}$$

Hence,

$$\mathbb{E}\left[\|M_{t,e}\|^2 \mid \mathcal{F}_{t,e}\right] = \sum_{k\in\mathcal{K}} \mathbb{E}\left[\|R^{(k)}\varepsilon_{t,e}^{(k)}\|^2 \mid \mathcal{F}_{t,e}\right].$$

Since $R^{(k)}P^{(k)} = R^{(k)}$ and the diagonal entries of $R^{(k)}$ lie in $[0,1]$, we have

$$\|R^{(k)}\varepsilon_{t,e}^{(k)}\|^2 = \|R^{(k)}P^{(k)}\varepsilon_{t,e}^{(k)}\|^2 \leq \|P^{(k)}\varepsilon_{t,e}^{(k)}\|^2.$$

Using Assumption (S2), we obtain

$$\begin{aligned}\mathbb{E}\big[\|M_{t,e}\|^2 \mid \mathcal{F}_{t,e}\big] &= \sum_{k\in\mathcal{K}} \mathbb{E}\Big[\|R^{(k)}\varepsilon_{t,e}^{(k)}\|^2 \mid \mathcal{F}_{t,e}\Big] \\ &\leq \sum_{k\in\mathcal{K}} \mathbb{E}\Big[\|P^{(k)}\varepsilon_{t,e}^{(k)}\|^2 \mid \mathcal{F}_{t,e}\Big] \\ &\leq |\mathcal{K}|\sigma^2.\end{aligned}$$

Since the client set is fixed, $|\mathcal{K}|$ is a constant independent of t, T, η, and E. Defining $C_R := |\mathcal{K}|$, and taking expectations on both sides, we get

$$\mathbb{E}\|M_{t,e}\|^2 \leq C_R\sigma^2.$$

Substituting this into the previous bound gives

$$\mathbb{E}\|A_t\|^2 \leq \frac{1}{E^2}\sum_{e=0}^{E-1} C_R\sigma^2 = C_R\frac{\sigma^2}{E}.$$

Bound on B_t. By Assumption (S1) and since each $R^{(k)}$ is diagonal with entries in $[0,1]$,

$$\|B_t\| \leq \frac{L}{E}\sum_{e=0}^{E-1}\sum_{k\in\mathcal{K}} \|\mathbf{w}_{t,e}^{(k)} - \mathbf{w}_t\|.$$

Applying Jensen's inequality gives

$$\|B_t\|^2 \leq \frac{L^2|\mathcal{K}|}{E}\sum_{e=0}^{E-1}\sum_{k\in\mathcal{K}} \|\mathbf{w}_{t,e}^{(k)} - \mathbf{w}_t\|^2.$$

Now,

$$\mathbf{w}_{t,e}^{(k)} - \mathbf{w}_t = -\eta\sum_{s=0}^{e-1} P^{(k)}g_{t,s}^{(k)},$$

so, again by Jensen,

$$\|\mathbf{w}_{t,e}^{(k)} - \mathbf{w}_t\|^2 \leq \eta^2 e\sum_{s=0}^{e-1}\|P^{(k)}g_{t,s}^{(k)}\|^2.$$

Taking expectations and using Assumption (S4),

$$\mathbb{E}\Big[\|\mathbf{w}_{t,e}^{(k)} - \mathbf{w}_t\|^2\Big] \leq \eta^2 e^2 G^2.$$

Therefore,

$$\mathbb{E}\big[\|B_t\|^2\big] \le \frac{L^2|\mathcal{K}|}{E}\sum_{e=0}^{E-1}\sum_{k\in\mathcal{K}}\eta^2 e^2 G^2 = \frac{L^2|\mathcal{K}|^2}{E}\eta^2 G^2\sum_{e=0}^{E-1} e^2$$
$$\le \frac{L^2|\mathcal{K}|^2}{E}\eta^2 G^2\sum_{e=0}^{E-1}(E-1)^2 = L^2|\mathcal{K}|^2\eta^2 G^2(E-1)^2.$$

Since $\mathcal{K}$ is fixed in the static regime, $|\mathcal{K}|^2$ is a constant. Renaming constants, we obtain

$$\mathbb{E}\big[\|B_t\|^2\big] \le C_B L^2\eta^2 G^2(E-1)^2,$$

for a constant $C_B > 0$ independent of t, T, η, and E.

Bound on C_t. By Assumption (S3), we have

$$\|C_t\|^2 = \left\|\sum_{k\in\mathcal{K}} R^{(k)}\nabla F^{(k)}(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\right\|^2 \le \zeta^2.$$

Therefore,

$$\mathbb{E}\big[\|C_t\|^2\big] \le \zeta^2.$$

Combining the three bounds proves the claim. □

Theorem E.2 (Convergence in the static case)**.** *Suppose Assumptions (S1)–(S4) hold, and let $\{\mathbf{w}_t\}_{t=0}^{T-1}$ be the sequence generated by the static training procedure above. Let*

$$\bar\eta := \eta E$$

and assume $\bar\eta \le 1/(8L)$. Then there exist constants $C_1, C_2, C_3 > 0$, independent of T, η, and E, such that

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla F(\mathbf{w}_t)\|^2\right] \le \frac{2(F(\mathbf{w}_0)-F_{\inf})}{\bar\eta T} + C_1 L\bar\eta\frac{\sigma^2}{E} + C_2 L^2\eta^2(E-1)^2G^2 + C_3\zeta^2,$$

where $F_{\inf}$ is a lower bound of F.

Equivalently, since $\bar\eta = \eta E$,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla F(\mathbf{w}_t)\|^2\right] \le \frac{2(F(\mathbf{w}_0)-F_{\inf})}{\eta E T} + C_1 L\eta\sigma^2 + C_2 L^2\eta^2(E-1)^2G^2 + C_3\zeta^2.$$

In particular, if E is fixed and $\eta = \Theta((E\sqrt{T})^{-1})$, then

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla F(\mathbf{w}_t)\|^2\right] = O(T^{-1/2} + \zeta^2).$$

In the exact-alignment case $\zeta = 0$, the standard nonconvex $O(T^{-1/2})$ stationarity rate is recovered.

Proof. By the module-wise aggregation rule introduced above, the global update can be written as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\sum_{e=0}^{E-1}\sum_{k\in\mathcal{K}} R^{(k)} g_{t,e}^{(k)}.$$

Define, as in Lemma E.1,

$$\bar g_t := \frac{1}{E}\sum_{e=0}^{E-1}\sum_{k\in\mathcal{K}} R^{(k)} g_{t,e}^{(k)}.$$

Then

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \bar\eta\bar g_t, \qquad \bar\eta = \eta E.$$

Using the decomposition from Lemma E.1, we can write

$$\bar g_t = \nabla F(\mathbf{w}_t) + A_t + B_t + C_t.$$

Therefore,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \bar{\eta}\big(\nabla F(\mathbf{w}_t) + A_t + B_t + C_t\big).$$

Let

$$G_t := \nabla F(\mathbf{w}_t).$$

By L-smoothness of F, the descent lemma gives

$$\begin{aligned} F(\mathbf{w}_{t+1}) &\le F(\mathbf{w}_t) + \langle G_t, \mathbf{w}_{t+1} - \mathbf{w}_t\rangle + \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\ &= F(\mathbf{w}_t) - \bar{\eta}\,\langle G_t, G_t + A_t + B_t + C_t\rangle + \frac{L\bar{\eta}^2}{2}\|G_t + A_t + B_t + C_t\|^2. \end{aligned}$$

We now take expectations. First, we have

$$\mathbb{E}\left[\langle G_t, A_t\rangle\right] = 0,$$

because $G_t = \nabla F(\mathbf{w}_t)$ is fixed at the beginning of round t, while A_t is a sum of conditionally zero-mean minibatch-noise terms, as shown in Lemma E.1. Hence,

$$\begin{aligned} \mathbb{E}[F(\mathbf{w}_{t+1})] \le \mathbb{E}[F(\mathbf{w}_t)] - \bar{\eta}\mathbb{E}\|G_t\|^2 - \bar{\eta}\mathbb{E}\left[\langle G_t, B_t + C_t\rangle\right] \\ + \frac{L\bar{\eta}^2}{2}\mathbb{E}\|G_t + A_t + B_t + C_t\|^2. \end{aligned}$$

We bound the remaining inner product using Cauchy–Schwarz and Young's inequality:

$$-\langle G_t, B_t + C_t\rangle \le \frac{1}{4}\|G_t\|^2 + \|B_t + C_t\|^2.$$

Moreover,

$$\|B_t + C_t\|^2 \le 2\|B_t\|^2 + 2\|C_t\|^2.$$

and

$$\|G_t + A_t + B_t + C_t\|^2 \le 4\left(\|G_t\|^2 + \|A_t\|^2 + \|B_t\|^2 + \|C_t\|^2\right).$$

Substituting these inequalities gives

$$\begin{aligned} \mathbb{E}[F(\mathbf{w}_{t+1})] \le \mathbb{E}[F(\mathbf{w}_t)] - \bar{\eta}\mathbb{E}\|G_t\|^2 + \frac{\bar{\eta}}{4}\mathbb{E}\|G_t\|^2 \\ + 2\bar{\eta}\mathbb{E}\|B_t\|^2 + 2\bar{\eta}\mathbb{E}\|C_t\|^2 \\ + 2L\bar{\eta}^2\mathbb{E}\|G_t\|^2 + 2L\bar{\eta}^2\mathbb{E}\|A_t\|^2 + 2L\bar{\eta}^2\mathbb{E}\|B_t\|^2 + 2L\bar{\eta}^2\mathbb{E}\|C_t\|^2. \end{aligned}$$

Since $\bar{\eta} \le 1/(8L)$, we have $2L\bar{\eta} \le 1/4$. Therefore,

$$-\bar{\eta} + \frac{\bar{\eta}}{4} + 2L\bar{\eta}^2 \le -\frac{\bar{\eta}}{2}.$$

Thus,

$$\begin{aligned} \mathbb{E}[F(\mathbf{w}_{t+1})] \le \mathbb{E}[F(\mathbf{w}_t)] - \frac{\bar{\eta}}{2}\mathbb{E}\|G_t\|^2 \\ + 2L\bar{\eta}^2\mathbb{E}\|A_t\|^2 + (2\bar{\eta} + 2L\bar{\eta}^2)\mathbb{E}\|B_t\|^2 + (2\bar{\eta} + 2L\bar{\eta}^2)\mathbb{E}\|C_t\|^2. \end{aligned}$$

From Lemma E.1 and its proof, the three components satisfy

$$\mathbb{E}\|A_t\|^2 \le C_R\frac{\sigma^2}{E},$$

$$\mathbb{E}\|B_t\|^2 \le C_B L^2\eta^2(E-1)^2G^2,$$

$$\mathbb{E}\|C_t\|^2 \le \zeta^2.$$

Since $\bar{\eta} \le 1/(8L)$, the factors $2\bar{\eta} + 2L\bar{\eta}^2$ are bounded by a constant multiple of $\bar{\eta}$. Hence, for suitable constants $C_1', C_2', C_3' > 0$,

$$\mathbb{E}[F(\mathbf{w}_{t+1})] \le \mathbb{E}[F(\mathbf{w}_t)] - \frac{\bar{\eta}}{2}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|^2 + C_1'L\bar{\eta}^2\frac{\sigma^2}{E} + C_2'\bar{\eta}L^2\eta^2(E-1)^2G^2 + C_3'\bar{\eta}\zeta^2.$$

Rearranging terms gives

$$\frac{\bar{\eta}}{2}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|^2 \le \mathbb{E}[F(\mathbf{w}_t)] - \mathbb{E}[F(\mathbf{w}_{t+1})] + C_1' L\bar{\eta}^2\frac{\sigma^2}{E} + C_2'\bar{\eta}L^2\eta^2(E-1)^2G^2 + C_3'\bar{\eta}\zeta^2.$$

Summing over $t = 0, \dots, T-1$, we obtain

$$\frac{\bar{\eta}}{2}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|^2 \le \sum_{t=0}^{T-1}\left(\mathbb{E}[F(\mathbf{w}_t)] - \mathbb{E}[F(\mathbf{w}_{t+1})]\right) + TC_1' L\bar{\eta}^2\frac{\sigma^2}{E} + TC_2'\bar{\eta}L^2\eta^2(E-1)^2G^2 + TC_3'\bar{\eta}\zeta^2.$$

The first sum is telescopic. Since $F(\mathbf{w}_0)$ is deterministic and $F(\mathbf{w}_T) \ge F_{\inf}$, we get

$$\frac{\bar{\eta}}{2}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|^2 \le F(\mathbf{w}_0) - F_{\inf} + TC_1' L\bar{\eta}^2\frac{\sigma^2}{E} + TC_2'\bar{\eta}L^2\eta^2(E-1)^2G^2 + TC_3'\bar{\eta}\zeta^2.$$

Dividing by $\bar{\eta}T/2$ yields

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|^2 \le \frac{2(F(\mathbf{w}_0) - F_{\inf})}{\bar{\eta}T} + 2C_1' L\bar{\eta}\frac{\sigma^2}{E} + 2C_2' L^2\eta^2(E-1)^2G^2 + 2C_3'\zeta^2.$$

Renaming constants proves the first claim.

Using $\bar{\eta} = \eta E$, this can equivalently be written as

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|^2 \le \frac{2(F(\mathbf{w}_0) - F_{\inf})}{\eta E T} + C_1 L\eta\sigma^2 + C_2 L^2\eta^2(E-1)^2G^2 + C_3\zeta^2.$$

If E is fixed and $\eta = \Theta((E\sqrt{T})^{-1})$, then the first two terms are $O(T^{-1/2})$, the third term is $O(T^{-1})$, and the fourth term remains proportional to ζ^2. Therefore,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|^2 = O(T^{-1/2} + \zeta^2).$$

In particular, when $\zeta = 0$, the standard $O(T^{-1/2})$ nonconvex stationarity rate is recovered. □

Empirical assessment of the alignment mismatch. To characterize the effect of module-wise aggregation, we compare its update direction with a sample-weighted average of the local gradients, evaluated at the same broadcast model w_t. For this comparison, we define the sample-weighted pooled objective

$$F_{\text{pool}}(w) := \sum_{k\in\mathcal{K}} p^{(k)} F^{(k)}(w), \qquad p^{(k)} := \frac{n^{(k)}}{\sum_{r\in K} n^{(r)}}.$$

At the beginning of each communication round, every client computes its gradient at the broadcast model w_t for this diagnostic. We then construct the module-wise aggregated direction

$$d_t = \sum_{k\in\mathcal{K}} R^{(k)}\nabla F^{(k)}(w_t)$$

and measure its empirical mismatch with the sample-weighted average of the local gradients, given by $\nabla F_{\text{pool}}(w_t)$:

$$\widehat{\zeta}_t^2 = \left\|d_t - \nabla F_{\text{pool}}(w_t)\right\|^2.$$

We evaluate $\widehat{\zeta}_t^2$ over the first 30 communication rounds, using three random seeds and one local epoch per round. We additionally measure the cosine similarity between d_t and $\nabla F_{\text{pool}}(w_t)$ to distinguish directional disagreement from differences in gradient magnitude.

Table 19 shows that the mismatch remains stable along the evaluated optimization trajectories, ranging from 0.0186 to 0.0203 on ASIA and from 0.0287 to 0.0290 on ALARM. In both cases, the maximum occurs at the first round and the minimum at round 30, indicating a mild decrease over the

Table 19: **Empirical alignment mismatch.** Empirical mismatch $\widehat{\zeta}_t^2$ between the module-wise aggregated direction and the gradient of the pooled objective over the first 30 communication rounds. The round attaining the minimum and maximum mismatch is reported in parentheses. Results are aggregated over three random seeds.

Dataset	Mean $\widehat{\zeta}_t^2$	Min. $\widehat{\zeta}_t^2$ (round)	Max. $\widehat{\zeta}_t^2$ (round)	Mean cosine sim.
ASIA	0.0194	0.0186 (30)	0.0203 (1)	0.949 ± 0.013
ALARM	0.0288	0.0287 (30)	0.0290 (1)	0.883 ± 0.007

evaluated rounds. Moreover, mean cosine similarities of 0.949 and 0.883, respectively, show that the module-wise aggregate remains strongly positively aligned with the pooled gradient.

Extension to dynamic environments with piecewise-stationary changes. We consider a piecewise-stationary dynamic process that can be decomposed into a sequence of static regimes, each corresponding to the interval between two consecutive change points. Within each regime s, we retain the participation model of Theorem E.2, with a fixed client set $\mathcal{K}_s$. Changes in the client set, supervision pattern, concept graph, or architecture occur only at regime boundaries.

We assume that the sequence of client and structural changes eventually stabilizes. In particular, given finite sets of task-relevant concepts and admissible dependencies, we assume that the aggregated structure eventually stabilizes and that the total number of regimes remains bounded independently of the training horizon. At a change point, the active client set, the available supervision, the concept graph, the induced objective, and the model architecture may change. The current model is then transferred to the parameter space of the new optimization problem and used to initialize the next regime. More formally, let

$$0 = \tau_0 < \tau_1 < \cdots < \tau_S = T$$

denote the regime boundaries, with changes occurring at $\tau_1, \ldots, \tau_{S-1}$. Within each interval

$$I_s := [\tau_s, \tau_{s+1}),$$

all quantities defining the optimization problem remain fixed. In particular, the model has a fixed parameter space $\mathcal{W}_s$ and a differentiable objective

$$F_s : \mathcal{W}_s \to \mathbb{R}.$$

Let

$$T_s := \tau_{s+1} - \tau_s$$

denote the duration of regime s. We assume that the regime-specific counterparts of Assumptions (S1)–(S4) hold within each regime, with constants uniformly bounded across regimes. For notational simplicity, let $L, \sigma, G, C_1, C_2, C_3$ denote common finite upper bounds on the corresponding regime-specific quantities. We further assume that every trainable module in regime s is supervised by at least one client in $\mathcal{K}_s$; retained modules with no current supervision are treated as fixed and excluded from the optimization variables of that regime.

At a transition time τ_s, the parameter space, objective, and supervision pattern may differ from those of the preceding regime. We denote by $w_{\tau_s}^-$ the parameters immediately before the transfer and by

$$w_{\tau_s}^+ \in \mathcal{W}_s$$

the post-transfer parameters used to initialize regime s. For notational uniformity, we set $w_{\tau_0}^+ := w_0$ and write $w_{\tau_s} := w_{\tau_s}^+$ for all s.

For each regime, define the expected regime-start objective gap as

$$\Gamma_s := \mathbb{E}[F_s(w_{\tau_s}) - F_{s,\inf}],$$

where $F_{s,\inf}$ is a lower bound on F_s. For $s \geq 1$, Γ_s measures the quality of the transferred initialization under the newly active objective.

Applying the proof of Theorem E.2 within each regime gives:

$$\frac{1}{T_s}\sum_{t=\tau_s}^{\tau_{s+1}-1}\mathbb{E}\Big[\|\nabla F_s(w_t)\|^2\Big] \le \frac{2\Gamma_s}{\bar{\eta}T_s} + C_1 L\bar{\eta}\frac{\sigma^2}{E} + C_2 L^2\eta^2(E-1)^2G^2 + C_3\zeta_s^2,$$

where $\bar{\eta} := \eta E \le 1/(8L)$ and ζ_s^2 denotes the module-wise alignment mismatch in regime s.

Multiplying the bound for regime s by T_s/T and summing over all regimes yields

$$\begin{aligned}\mathcal{G}_T &:= \frac{1}{T}\sum_{s=0}^{S-1}\sum_{t=\tau_s}^{\tau_{s+1}-1}\mathbb{E}\Big[\|\nabla F_s(w_t)\|^2\Big] \\ &\le \frac{2}{\bar{\eta}T}\sum_{s=0}^{S-1}\Gamma_s + C_1 L\bar{\eta}\frac{\sigma^2}{E} \\ &\quad + C_2 L^2\eta^2(E-1)^2G^2 + \frac{C_3}{T}\sum_{s=0}^{S-1}T_s\zeta_s^2.\end{aligned}$$

Here, $\mathcal{G}_T$ measures average stationarity over the complete dynamic trajectory, evaluating each iterate with respect to the objective active at that round.

By assumption, the number of regimes is bounded independently of T, i.e.,

$$S \le S_{\max}$$

for some constant $S_{\max}$. Assume further that $\Gamma_0 = O(1)$ and that the post-transition initialization gaps satisfy

$$\Gamma_s \le B, \qquad s \ge 1.$$

It follows that

$$\sum_{s=0}^{S-1}\Gamma_s \le \Gamma_0 + (S-1)B \le \Gamma_0 + (S_{\max}-1)B = O(1).$$

With fixed E and the same step-size choice as in Theorem E.2,

$$\eta = \Theta\left(\frac{1}{E\sqrt{T}}\right), \qquad \bar{\eta} = \Theta(T^{-1/2}),$$

the contribution of the regime-start objective gaps is $O(T^{-1/2})$. If the module-wise alignment mismatch is uniformly bounded across regimes, i.e.,

$$\zeta_s^2 \le \zeta^2 \qquad \text{for all } s,$$

then, similarly to before, we obtain

$$\mathcal{G}_T = O\Big(T^{-1/2} + \zeta^2\Big).$$

Thus, under the stated assumptions, the piecewise-stationary dynamic trajectory satisfies the same average-stationarity guarantee as the static case, up to the module-wise alignment mismatch. Under exact alignment in every regime, i.e., $\zeta_s = 0$ for all s, we recover

$$\mathcal{G}_T = O(T^{-1/2}).$$

The assumptions above are natural in settings where new institutions, concepts, and dependencies are introduced during a bounded onboarding or adaptation phase, after which the federation stabilizes. In continuously evolving deployments, the same piecewise analysis suggests that the stationarity bound would additionally depend on the frequency and cumulative cost of regime changes. Extending the analysis to persistent non-stationarity, for example through explicit control of these quantities, is an interesting direction for future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Limitations and future work paragraph in section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See subsection E.2.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See section 4, Appendix A, Appendix B, and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The data used is publicly available, and we provide the official code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See section 4, Appendix A, Appendix B, and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See figures and tables.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See subsection C.3 for the hardware configuration and subsection D.5 for the computational complexity analysis, measured runtime overheads, and total training times for representative configurations (Table 5). Exact total compute across all experimental runs was not systematically tracked.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have thoroughly reviewed and adhered to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See section 4 and section 6.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not address safeguards for the responsible release of high-risk data or models, as our research does not involve such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See subsection C.3 for software, dataset, and pretrained-model licenses and usage conditions, and Appendix A for dataset provenance and preparation. The original assets are credited throughout the paper, and our implementation is released under the MIT license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: See section 4, Appendix C, and the provided code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with user study.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.