FedCAG: Federated Causality-Aware Graph Learning for Multi-Cloud Workload Forecasting

Yongcan Luo[1] **Zhengjie Yang**[1,*] **Jiahao Zheng**[1] **Hao Wang**[1] **Wei Bao**[2] **Dapeng Wu**[1]

[1]City University of Hong Kong [2]The University of Sydney

Abstract

Recent large-scale outages at major cloud providers such as AWS and GCP have exposed the fragility of relying on a single cloud. To improve resilience, fault tolerance, and business continuity, many enterprises are moving their services to multi-cloud environments. However, multi-cloud deployment also makes workload forecasting substantially harder. Existing multi-cloud workload forecasting methods are typically designed for either centralized or isolated environments, leading to risks of private data leakage or limited generalization. To address these challenges, we propose **FedCAG**, a **Fed**erated **C**ausality-**A**ware **G**raph learning paradigm for multi-cloud workload forecasting. Instead of treating federated learning as parameter averaging over local predictors, FedCAG jointly federates predictive models and graph-structured dependency priors. Each client constructs causality-aware, spatial, and temporal graphs from its private telemetry to model directed inter-metric influence, metric-level interactions, and intra-window temporal dynamics. To address strong cross-cloud heterogeneity, FedCAG further combines causality-aware representation fusion, adaptive graph refinement, and client-specific personalization, enabling the global model to benefit from shared workload structures while preserving local specificity. Experiments on real-world datasets show that FedCAG consistently outperforms mainstream federated baselines and even several centralized methods trained on the full dataset, delivering stable and accurate forecasting across heterogeneous and privacy-constrained deployments.

1 Introduction

Recent large-scale outages at major cloud service providers (e.g., AWS and GCP) have exposed the fragility of relying on a single cloud. These outages caused prolonged service disruptions and affected government services, financial systems, and other daily applications. To improve resilience, fault tolerance, and business continuity, many enterprises are migrating from single-cloud deployment to distributed multi-cloud environments [1, 2]. However, multi-cloud deployment also makes system management and workload forecasting more difficult. In a multi-cloud system, workloads run on heterogeneous virtual machines across different providers, regions, and hardware configurations. As a result, key workload indicators, such as CPU utilization, memory usage, and network traffic, often show strong temporal non-stationarity, bursty variations, and complex cross-metric coupling [3–5]. These properties make multi-cloud workload dynamics more volatile and structurally heterogeneous than those in single-cloud environments, posing challenges for accurate workload forecasting [6].

Existing workload forecasting methods are mainly designed for either centralized or isolated environments, as illustrated in Figure 1 (left). Centralized approaches collect fine-grained telemetry from all sites and train a single forecasting model. Typical examples include deep sequence models and spatial-temporal graph models [7–9]. This paradigm can exploit global information, but it assumes

*Corresponding author.

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

that raw workload traces can be freely shared across cloud providers, tenants, and regions. Such an assumption is often unrealistic in multi-cloud deployments. The main obstacles include privacy regulations, data sovereignty requirements, provider-specific data ownership, and cross-organization governance constraints [10, 11]. On the other side, isolated approaches train a separate local model for each VM, site, or cloud provider [12, 5]. These methods avoid raw-data sharing. However, they cannot use transferable workload patterns across infrastructures. As a result, they often suffer from limited generalization under rare bursts, distribution shifts, and highly heterogeneous client data. Thus, neither centralized nor isolated forecasting provides a satisfactory foundation for privacy-preserving and generalizable multi-cloud workload prediction.

Federated Learning (FL) [13] offers a natural alternative for privacy-preserving workload forecasting. FL enables multiple clients to collaboratively train a model without transmitting raw telemetry [14]. In a typical FL workflow, each cloud site or VM optimizes a local model, and a server aggregates model parameters to obtain a global predictor. However, conventional FL is still insufficient for multi-cloud workload forecasting. The challenge is not limited to non-IID client data. Workload evolution is also shaped by structured dependencies among system indicators, such as lead-lag effects between network traffic and CPU usage. Most federated forecasting methods remain parameter-centric: they aggregate neural predictors, but rarely learn, communicate, or personalize the dependency structures that govern local workload dynamics [15, 16]. In multi-cloud environments, provider-specific workloads induce distribution shifts across clients [17], and temporal patterns vary across regions, VM types, and resource provisioning policies. Naive parameter averaging may therefore pull local models away from client-specific optima [18]. Meanwhile, the structural relationships among workload indicators are often ignored, predetermined, or treated as local preprocessing artifacts [19]. Conventional FL methods, therefore, lack a principled mechanism to adapt both model parameters and structural dependencies across heterogeneous cloud clients.

To address these challenges, we propose **FedCAG**, a **Fed**erated **C**ausality-**A**ware **G**raph learning algorithm for multi-cloud workload forecasting. FedCAG introduces a new graph-native federated learning paradigm that goes beyond parameter-centric FL. Instead of only averaging local predictors, FedCAG jointly federates predictive models and graph-structured dependency priors. Each client constructs temporal, spatial, and causality-aware graphs from private telemetry to model intra-window dynamics, metric-level interactions, and directed lead-lag dependencies. These graphs are learned locally, aggregated at the server to extract transferable structural priors, and personalized back to each client to preserve local workload behavior. FedCAG further combines a dual-graph encoder, causality-aware fusion, adaptive edge refinement, and lightweight client-private adapters to handle temporal heterogeneity, unstable dependencies, and client drift. By federating dynamic structural knowledge together with model parameters, FedCAG differs fundamentally from conventional federated forecasting methods. The main contributions of this work are summarized as follows:

- We first formulate multi-cloud workload forecasting as a privacy-preserving federated multivariate time-series prediction problem and propose FedCAG, a new graph-native federated learning paradigm that jointly federates model parameters and graph-structured dependency priors, moving beyond parameter-centric FL for workload forecasting.
- We develop a causality-aware dual-graph neural architecture that separately models intra-window temporal dynamics and directed inter-metric influence, enabling structured representation learning beyond correlation-based sequence modeling.
- We design a federated graph aggregation and personalization mechanism that shares transferable structural priors across cloud providers while preserving client-specific workload characteristics through adaptive graph refinement and private adapters.
- We conduct extensive experiments on real-world multi-cloud workloads, comparing FedCAG with 15 centralized, federated, and graph-based baselines and demonstrating robust forecasting performance under heterogeneous and privacy-constrained deployments.

2 Related Work

Workload forecasting has long been a fundamental problem in large-scale distributed computing systems, as accurate predictions directly support resource allocation, autoscaling, and service-level management. Prior research spans a broad spectrum, ranging from early centralized statistical models to modern deep learning architectures, and more recently to privacy-preserving federated learning

frameworks. In Appendix A, we review relevant literature along three dimensions: (i) centralized time-series prediction for cloud workloads, (ii) federated learning approaches for time-series and spatio-temporal forecasting, and (iii) federated learning more broadly from systems and application perspectives. We highlight both the strengths and limitations of these approaches, particularly under non-IID and multi-cloud settings, which motivate the design choices of our proposed framework.

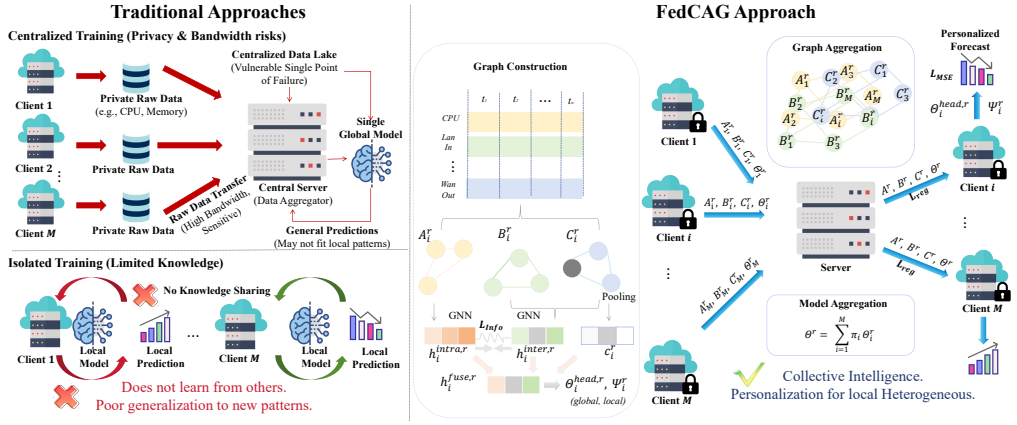


Figure 1: Overview of FedCAG (right) vs. traditional approaches (left). Traditional approaches suffer from data privacy and bandwidth risks (centralized training) or limited knowledge and poor generalization to new patterns (isolated training). FedCAG addresses these issues by developing a causality-inspired federated dual-graph neural forecasting method tailored for non-IID multi-cloud workloads. It achieves strong generalization while preserving client-specific characteristics.

3 Methodology

3.1 Problem Formulation

We study multi-metric workload forecasting in distributed (multi-cloud) environments under strict data locality constraints (non-IID). We consider M clients indexed by $i \in \{1, \ldots, M\}$, where each client corresponds to a cloud site, tenant, or VM group. Client i observes a private multivariate time series X_i^t, where $X_i^t \in \mathbb{R}^n$ is client i's n-dimensional vector of system metrics (e.g., CPU usage, memory, and network I/O) at time t. Given a sliding window length w, the local forecasting task is to predict the next-step workload X_i^{t+1} from the historical window $X_i^{t-w+1:t} \triangleq \{X_i^{t-w+1}, \ldots, X_i^t\}$. We address this task in a federated learning setting, where clients jointly train a forecasting model without sharing raw data. The ultimate goal is to learn a robust global model that generalizes across heterogeneous clients while enabling client-specific adaptation under non-IID workloads.

3.2 Overview of FedCAG

We propose **FedCAG**, a causality-inspired federated dual-graph neural forecasting algorithm for multi-cloud workloads illustrated in Figure 1 (right). New technologies (e.g., reliability weights and exponential moving average smoothing for graph aggregation, convex blending for global priors transfer and local priors specificity, and a trainable and lightweight private adapter Ψ_i for preserving clients' unique local workload characteristics) are developed, all for stable and accurate workload forecasting under non-IID and privacy-constrained multi-cloud environments. The key notations are summarized in Table 10 in the Appendix.

3.3 Client-side Operations

Graph Construction. Multi-cloud telemetry exhibits heterogeneous dependencies (e.g., across timestamps within a VM or across metrics and VMs). This complexity is further amplified under non-IID multi-cloud deployments. As a result, typical workload forecasting methods [20, 21] that rely on a single adjacency matrix or learn multiple graphs under the same inductive bias cannot

precisely capture these dependencies, ultimately degrading forecasting performance. To address this, in FedCAG, we develop three complementary graphs, each targeting a distinct source of structure.

At the beginning of each communication round $r \in [1, R]$, for each client i, we first construct a temporal graph $\mathbf{A}_i^r$ to model short-term temporal adjacency within an input window $X_i^{t-w+1:t}$ on a specific VM. This graph captures local dynamics such as queuing delays, warmup/cooldown, and autoscaling transients that primarily govern autoregressive evolution. We then zoom out to a broader scope and develop a spatial graph $\mathbf{B}_i^r$ to capture undirected co-dependency among metrics that are functionally coupled by shared resource contention and system design (e.g., CPU, memory, and I/O interactions). Such inter-metric dependencies across VMs provide a stable interaction backbone. Third, since $\mathbf{B}_i^r$ encodes undirected metric affinity, it enforces symmetric message passing, which may introduce backward information leakage (i.e., propagating effects back to their potential causes) and cannot represent lead-lag effects [22]. To this end, we further develop a causality graph $\mathbf{C}_i^r$ to encode directed predictive influence across metrics, specifically tailored to capture lead-lag effects, such as sustained WAN ingress preceding CPU pressure or memory backpressure. Mathematically, for each client i, we construct the window of time series $X_i^{t-w+1:t}$ into the three graphs as $\mathbf{A}_i^r \in \mathbb{R}^{w\times w}$, $\mathbf{B}_i^r \in \mathbb{R}^{n\times n}$, and $\mathbf{C}_i^r \in \mathbb{R}^{n\times n}$, respectively. The detailed graph construction process, including sparsification, is provided in Appendix B.

Graph Personalization. After receiving the aggregated global graphs $\mathbf{A}^{r-1}, \mathbf{B}^{r-1}$, and $\mathbf{C}^{r-1}$ from the last communication round $r-1$, each client blends the global priors/graphs with its locally constructed graphs to obtain personalized structural priors:

$$\hat{\mathbf{A}}_i^r = \alpha_i^r \mathbf{A}^{r-1} + (1-\alpha_i^r)\mathbf{A}_i^r, \tag{1}$$

and similarly for $\hat{\mathbf{B}}_i^r$ and $\hat{\mathbf{C}}_i^r$. The learnable $\alpha_i \in [0,1]$ is tuned by Optuna [23] per communication round as the structural blending coefficient used to construct personalized priors before local training. α_i^r controls how strongly the client i follows the global structure: smaller α_i^r favors local specificity, while larger α_i^r promotes global transfer. We set α_i^r to maximize client-level validation performance by leveraging global structural priors when beneficial, while avoiding negative transfer for clients with distinctive non-IID dependency patterns.

Dual-Graph Encoding. After obtaining the personalized graphs, the next step is to encode the graphs to jointly model *intra-temporal dependencies* and *inter-metric dependencies* for a given input window $X_i^{t-w+1:t}$. Please note that in FedCAG, we explicitly decouple inter-metric influence from intra-window temporal propagation so that message passing respects the appropriate dependency type, while existing methods mess up all information from the same multivariate series under identical assumptions, leading to sub-optimal performance.

To encode intra-temporal dependencies, we input $\hat{\mathbf{A}}_i^r$ into a Graph Neural Network (GNN) [24] to obtain intra-temporal embeddings

$$h_i^{\text{intra},r} = \text{GNN}\left(\hat{\mathbf{A}}_i^r\right). \tag{2}$$

To encode inter-metric dependencies, we input both $\hat{\mathbf{B}}_i^r$ and $\hat{\mathbf{C}}_i^r$ into a GNN to obtain inter-metric embeddings

$$h_i^{\text{inter},r} = \text{GNN}\left(\hat{\mathbf{B}}_i^r, \hat{\mathbf{C}}_i^r\right). \tag{3}$$

Here, $\hat{\mathbf{C}}_i^r$ provides directional routing that biases propagation from potential drivers to responders, while $\hat{\mathbf{B}}_i^r$ serves as a structural support graph that preserves stable metric couplings and maintains connectivity when $\hat{\mathbf{C}}_i^r$ is sparse.

The two GNNs are trained in parallel for each client i at each local training epoch $e \in [1, E]$ within each communication round $r \in [1, R]$. For simplicity, we omit the local training epoch index e. The above **Dual-graph Encoder** produces rich and accurate intra-temporal and inter-metric embeddings, serving as representation inputs for downstream fusion and prediction. Additional GNN message passing details are provided in Appendix B.

Causality Fusion. To strengthen the impact of the personalized causality graph $\hat{\mathbf{C}}_i^r$ for a better capture of lead-lag effects, we further develop a causality descriptor that aligns with the two output embeddings. We first leverage a graph pooling operator $\phi(\cdot)$ to compact $\hat{\mathbf{C}}_i^r$ into the same dimension

as that in $h_i^{\text{intra},r}$ and $h_i^{\text{inter},r}$, denoted by c_i^r. Then, we fuse intra-temporal, inter-metric, and compacted causality information via a Multi-Head Attention (MHA) [25]:

$$h_i^{\text{fuse},r} = \text{MHA}\left(\left[h_i^{\text{intra},r} \| \, h_i^{\text{inter},r} \| \, c_i^r\right]\right), \tag{4}$$

where $\|$ denotes concatenation. Here, we utilize the concatenated representation as the input token set to allow the MHA to learn their joint interactions and cross-dependencies, rather than enforcing a fixed role assignment (e.g., assigning $h_i^{\text{intra},r}$, $h_i^{\text{inter},r}$, and c_i^r as Q, K, and V, respectively) that may not generalize across heterogeneous clients. By adopting MHA, FedCAG can perform dynamic and context-dependent fusion, enabling the model to reweight spatial, temporal, and causal cues differently across clients and time windows under non-IID workloads in multi-cloud environments.

Personalized Prediction. Although the previous operations may somewhat mitigate the negative effects of non-IID workloads across clients (graph processing and embedding fusion), we still need more effort to address the non-IID issue from a model-architecture perspective. To this end, we develop a client-private adapter Ψ_i^r to preserve client-specific model characteristics and provide personalized calibration.

As mentioned in Dual-Graph Encoding, two GNNs are trained in parallel in (2) and (3), parameterized by a backbone $\Theta_i^{\text{bb},r}$. We then attach a prediction head (a two-layer MLP), parameterized by $\Theta_i^{\text{head},r}$, to help predict the result. We denote the client i's model parameter at round r as $\Theta_i^r \triangleq (\Theta_i^{\text{bb},r}, \Theta_i^{\text{head},r})$. The preliminary result is predicted by $f(h_i^{\text{fuse},r}; \Theta_i^{\text{head},r})$, using $h_i^{\text{fuse},r}$ as the input and $\Theta_i^{\text{head},r}$ as the model parameters. To achieve client-specific calibration for more accurate predictions, we develop a lightweight two-layer MLP that outputs a residual correction to assist the preliminary prediction. We denote the result of the personalized residual correction as $g(h_i^{\text{fuse},r}; \Psi_i^r)$. Specifically, we conduct an element-wise addition for two predictions, followed by another two-layer MLP to reduce the dimension and obtain the final prediction result

$$\hat{X}_i^{t+1} = \text{MLP}\left(f\left(h_i^{\text{fuse},r}; \Theta_i^{\text{head},r}\right) \oplus g\left(h_i^{\text{fuse},r}; \Psi_i^r\right)\right). \tag{5}$$

Since $\Theta_i^{\text{head},r}$ is periodically aggregated and redistributed from the server every communication round, it captures provider-agnostic forecasting regularities. On the other hand, the private adapter Ψ_i^r is trained on the client's dataset and kept locally, leading to personalized calibration. In this way, both results contribute to the final prediction, thereby enhancing prediction performance and positively influencing the training process in non-IID environments.

Local Optimization. By gathering all above efforts, we develop the final objective loss, which contains three parts (communication round r and local epoch index e are omitted): (i) $\mathcal{L}_{\text{mse}} = \|\hat{X}_i^{t+1} - X_i^{t+1}\|_2^2$ for prediction among n-dimensional vectors of system metrics; (ii) $\mathcal{L}_{\text{info}}$, an InfoNCE loss to contrast temporal and spatial embeddings for consistent workload states; and (iii) $\mathcal{L}_{\text{reg}} = \|\Theta_i^r - \Theta^{r-1}\|_2^2$ for stabilized training under non-IID environments, preventing local models from drifting away from the global model. To summarize, the final loss is

$$\mathcal{L}_i = \mathcal{L}_{\text{mse}} + \lambda_1 \mathcal{L}_{\text{info}} + \lambda_2 \mathcal{L}_{\text{reg}}, \tag{6}$$

where λ_1 and λ_2 are adjustable hyperparameters. By utilizing this composite loss, FedCAG can balance forecasting accuracy, representation alignment, and federated stability, enhancing the algorithm generalization capability while maintaining personalization among clients in non-IID environments. The optimization details are provided in Appendix C.

3.4 Server-side Operations

After E local training epochs, each client uploads local graphs $\mathbf{A}_i^r, \mathbf{B}_i^r$, and $\mathbf{C}_i^r$ and local model parameters Θ_i^r to the server. Then, the server aggregates local values to obtain global structural priors $\mathbf{A}^r, \mathbf{B}^r$, and $\mathbf{C}^r$ and global model parameters Θ^r, which will be distributed back to each client for the next round of local training.

Graph Aggregation. To aggregate global structural priors, we first construct reliability-weighted global priors

$$\tilde{\mathbf{A}}^r = \text{Norm}\left(\sum_{i=1}^{M} \rho_i^r \, \mathbf{A}_i^r\right), \tag{7}$$

and similarly for $\tilde{\mathbf{B}}^r$ and $\tilde{\mathbf{C}}^r$. Here, $\mathrm{Norm}(\cdot)$ enforces bounded edge weights, and $\rho_i^r \in [0,1], \sum_{i=1}^M \rho_i^r = 1$ is a learnable parameter tuned by Optuna per communication round. Larger ρ_i^r demonstrates a more reliable client i's graph updates, so we let this client contribute more to the global structural priors, and vice versa. Then, the reliability-weighted global priors are smoothed using Exponential Moving Averages (EMA) to obtain the final global priors:

$$\mathbf{A}^r = \beta \mathbf{A}^{r-1} + (1-\beta)\tilde{\mathbf{A}}^r, \tag{8}$$

and similarly for $\mathbf{B}^r$ and $\mathbf{C}^r$. The EMA weight $\beta \in [0,1)$ is a hyperparameter that controls the stability and adaptivity trade-off.

Model Aggregation. Inspired by FedAvg [13], we leverage a weighted average to aggregate client model parameters to obtain the global model parameters

$$\Theta^r = \sum_{i=1}^{M} \pi_i \Theta_i^r, \tag{9}$$

where π_i is the weight/contribution of client i's model parameters corresponding to the number of training windows available at client i (number of training data samples), and $\sum_{i=1}^M \pi_i = 1$. The end-to-end training and inference of FedCAG is detailed in Appendix D.

3.5 Convergence Discussion

Due to page limitations, we present a skeleton of the convergence analysis. Unlike the backbone $\Theta_i^{\mathrm{bb},r}$ and prediction head $\Theta_i^{\mathrm{head},r}$ that are aggregated every communication round, each client's private adapter Ψ_i^r is only trained locally. We need a mechanism to align our convergence analysis with typical methods that aggregate the complete client parameters. To this end, we propose a virtual adapter aggregation that can be aggregated and redistributed. The update rule of the virtual adapter is

$$\Psi^r = \sum_{i=1}^{M} \pi_i \Psi_i^r. \tag{10}$$

Then, we obtain the virtual global model parameters

$$\hat{\Theta}^r \triangleq (\Theta^{\mathrm{bb},r}, \Theta^{\mathrm{head},r}, \Psi^r). \tag{11}$$

Since the gradient is backpropagated through each model component in each local epoch e within each communication round r, we can derive that the virtual global model is iterated as

$$\hat{\Theta}^r = \hat{\Theta}^{r-1} - \eta \sum_{i=1}^{M} \pi_i \sum_{e=1}^{E} \left[\nabla_{\Theta^{\mathrm{bb}}} \mathcal{L}_i(\Theta_i^{r,e}), \nabla_{\Theta^{\mathrm{head}}} \mathcal{L}_i(\Theta_i^{r,e}), \nabla_{\Psi} \mathcal{L}_i(\Theta_i^{r,e})\right], \tag{12}$$

where η is the learning rate and $\nabla_{\Theta^{\mathrm{bb}}} \mathcal{L}_i$, $\nabla_{\Theta^{\mathrm{head}}} \mathcal{L}_i$, and $\nabla_{\Psi} \mathcal{L}_i$ are gradients related to Θ^{bb}, Θ^{head}, and Ψ respectively. Mathematically, we can treat the combined gradients as a whole, which can be directly aligned with typical convergence analyses [26–28]. Since the final global model Θ^R is within $\hat{\Theta}^R$, we prove the convergence. More details are in Appendix N.

4 Experiments

4.1 Experimental Setup

Datasets & Baselines. We evaluate FedCAG on real-world cloud workload traces collected from the MUCEP platform, including CPU, memory, and network traffic time series data. More details about the dataset and MUCEP can be found in Appendix E. To systematically analyze scalability and cross-provider heterogeneity, we construct four datasets with increasing size and diversity. (i) *Single Cloud 1:* contains 5 Alibaba Cloud VMs and represents a small and relatively homogeneous environment. (ii) *Single Cloud 2:* expands this setting to 10 Alibaba Cloud VMs, increasing dimensionality while preserving provider consistency. (iii) *Mixed Cloud 1:* consists of 10 VMs evenly split between Alibaba Cloud and Tencent Cloud, introducing cross-provider heterogeneity while keeping the number of clients fixed. (iv) *Mixed Cloud 2:* further increases both scale and

Table 1: Average MAE and RMSE across 10 runs. In each configuration, centralized baselines train on the pooled union of all participating VMs' training data; federated methods retain each VM's data locally.

Methods	Settings	Single Cloud 1		Single Cloud 2		Mixed Cloud 1		Mixed Cloud 2	
		MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
Arima [29]	Centralized	1.3351	2.3570	1.5036	2.7131	1.6432	2.9539	1.9012	3.5042
GRU [30]	Centralized	1.1582	2.1458	1.3812	2.4532	1.4932	2.6821	1.7832	3.0148
LSTM [31]	Centralized	1.0539	1.9071	1.2290	2.2043	1.3823	2.7005	1.5371	2.8324
Autoformer [7]	Centralized	0.8825	1.7058	1.0296	1.9553	1.1347	2.3401	1.2516	2.5525
GPT4TS [8]	Centralized	0.8375	1.6628	0.9874	1.9241	1.1023	2.2139	1.2241	2.4024
HGCN [9]	Centralized	0.9268	1.7845	1.1414	2.0837	1.1983	2.1994	1.4584	2.6534
STSGCN [19]	Centralized	0.9011	1.7593	1.0563	2.0595	1.1204	2.1891	1.4121	2.5748
STPGNN [22]	Centralized	0.8166	1.6272	0.9581	1.8865	1.0842	2.1642	1.2142	2.3592
FedGRU [32]	Federated	1.5423	2.8423	1.8423	3.2349	1.8903	3.3214	2.3584	4.8323
FL-LSTM [33]	Federated	1.4498	2.7492	1.8104	3.1032	1.8591	3.1842	2.3391	4.7998
FASTGNN [34]	Federated	1.4203	2.7628	1.6760	3.0048	1.7167	3.0848	2.1469	4.5848
FedSTN [16]	Federated	1.2892	2.5742	1.5213	2.9861	1.5613	2.8661	1.9913	4.3661
CNFGNN [35]	Federated	1.2139	2.4199	1.4324	2.8071	1.4624	2.8172	1.9024	3.1871
UFCL [36]	Federated	1.1048	2.0132	1.3037	2.3353	1.3437	2.4153	1.7737	2.7153
FEDCroST [18]	Federated	0.9893	1.8423	1.1674	2.1371	1.1974	2.2171	1.6374	2.6179
FedCAG	Federated	**0.9241**	**1.7545**	**1.0425**	**2.0119**	**1.0512**	**2.0421**	**1.4228**	**2.5841**

heterogeneity by combining 10 Alibaba Cloud and 5 Tencent Cloud VMs to illustrate a real-world heterogeneous multi-cloud scenario. The implementation details are provided in Appendix F. We compare FedCAG against two categories of baselines: 8 centralized algorithms covering classical statistical models, deep sequence models, and spatio-temporal GNNs, and 7 federated algorithms covering sequence-based and graph-based FL predictors designed to handle privacy constraints. Baselines are cited in Table 1 and their details are provided in Appendix G. More experimental results can be found in the Appendix, including the dropout study Appendix H, the personalization analysis Appendix I, and the inference time comparison Appendix J.

4.2 Main Results

Performance Comparison. Table 1 reports forecasting performance under four deployment configurations, spanning homogeneous single-cloud settings to large mixed-provider environments. As the number of VMs grows from 5 (*Single Cloud 1*) to 10 (*Single Cloud 2* and *Mixed Cloud 1*) and further to 15 (*Mixed Cloud 2*), both data dimensionality and workload diversity increase, enabling us to assess scalability and robustness under heterogeneity. The centralized comparison measures accuracy under pooled data access versus distributed training and different model architectures; it does not isolate the benefit of federation or equalize client-specific adaptation. At 50 VMs, FedCAG achieves MAE/RMSE of 2.4013/4.8304 and remains the strongest evaluated federated method. Appendix K gives that comparison and the run-to-run standard deviations for selected methods on *Mixed Cloud 2*. Appendix M reports the training-time breakdown.

Centralized models scale well from 5 to 10 homogeneous clients. With more training samples, temporal models such as GPT4TS, Autoformer, and STPGNN capture long-range dependencies and remain competitive. However, when multiple providers are mixed, the gains from additional samples diminish. Mixed-provider workloads introduce inconsistent CPU bursting, distinct I/O patterns, and varying network interference, leading to higher errors even with full access to raw data. Transformer-based methods are especially sensitive, as attention implicitly assumes a shared temporal structure that weakens when periodicities diverge across providers. Centralized graph-based models are slightly more robust due to spatio-temporal inductive priors, but still degrade in mixed settings, highlighting the challenge of learning a single global adjacency under heterogeneous metric relationships.

Federated methods show a more complex scaling trend. Even within single-provider settings, increasing client count introduces non-IIDness because VMs differ in arrival patterns, memory profiles, and network volatility. Methods based on simple parameter averaging, such as FedGRU

and FedLSTM, degrade noticeably before cross-provider heterogeneity appears. When clients span different providers, optimization conflicts intensify: local minima diverge, client gradients misalign, and global updates become less consistent. Federated graph approaches (FASTGNN, FEDSTN, CNFGNN, and FEDCroST) partially alleviate this issue via structural priors or personalization, but still weaken as diversity increases. In contrast, FedCAG remains more stable as scale and heterogeneity grow, and achieves performance comparable to centralized models (similar performance with HGCN).

Non-IID Data Analysis. A revealing comparison is made between *Single Cloud 2* and *Mixed Cloud 1*, which share the same number of clients but differ in distribution. *Single Cloud 2* contains VMs from one provider with similar scheduling and resource management, allowing centralized models to preserve temporal structure and cross-metric correlations. In *Mixed Cloud 1*, provider-specific architectures and I/O regimes disrupt these relationships, causing an immediate accuracy drop even for centralized training, which reflects the difficulty of learning a unified representation from mismatched temporal and spatial patterns.

On the other hand, federated methods show a smaller decline from *Single Cloud 2* to *Mixed Cloud 1*. Since some FL baselines have already supported client-level adaptation, they rely less on a single shared distribution, and personalization helps absorb cross-provider shifts. FedCAG is the most stable among federated methods, as directed causal graphs provide transferable structures across providers, while the personalized adapter preserves client-specific workload signatures without sacrificing global consistency.

4.3 Ablation Studies

Table 2 summarizes the ablation results on *Mixed Cloud 2*, which represents the most heterogeneous and challenging setting with 15 VMs across multiple providers. We focus on graph design and loss functions. Appendix L further ablates graph-prior sharing, graph personalization, reliability weighting, EMA smoothing, fusion, and the private adapter.

Table 2: Ablation Results on Mixed Cloud 2 (MAE / RMSE).

Ablation	Variant	MAE	RMSE
Graph design	w/o $\mathbf{C}_i^r$	1.4564	2.6445
	w/o $\mathbf{A}_i^r$	1.4739	2.6827
	w/o $\mathbf{B}_i^r$	1.4606	2.6653
InfoNCE	w/o $\mathcal{L}_{\text{info}}$	1.4428	2.6115
	w/o $\mathcal{L}_{\text{reg}}$	1.4480	2.6312

Graph Design. Removing any graph component leads to consistent performance degradation. Eliminating the temporal graph $\mathbf{A}_i^r$ results in the largest increase in error, indicating that the short-range temporal structure within each window remains important under strong heterogeneity. Removing either the spatial graph $\mathbf{B}_i^r$ or the causality graph $\mathbf{C}_i^r$ also degrades accuracy, suggesting that spatial and causal information provide complementary but partially substitutable cues. These results indicate that the full dual-graph design in FedCAG is the most effective when cross-metric dependencies vary significantly across VMs and providers.

Loss Function Components. Disabling InfoNCE alignment ($\lambda_1 = 0$) slightly worsens performance, showing that moderate contrastive alignment helps coordinate intra-temporal and inter-metric representations without over-constraining the model. In addition, removing the federated regularizer $\mathcal{L}_{\text{reg}}$ ($\lambda_2 = 0$) also degrades performance, indicating that the FedProx-style constraint is important for preventing excessive local drift and improving stability under severe non-IID workloads.

4.4 Hyperparameter Sensitivity Analysis

To understand how modeling and federated design choices affect prediction performance, we conduct a hyperparameter sensitivity analysis on six key hyperparameters of FedCAG: (i) the window length w used for graph construction and prediction, (ii) the β controls the adaptivity of global priors, (iii) the InfoNCE weight λ_1, (iv) the regulation coefficient λ_2 (inspired by FedProx [26]), (v) the Gumbel temperature τ that regulates adaptive edge gating, and (vi) the hidden dimension of the graph encoder d_{hid}. For each hyperparameter, we sweep over a small grid while fixing the others at their default values ($w = 24, \beta = 0.85, \lambda_1 = 0.3, \lambda_2 = 0.01, \tau = 0.5$, and $d_{\text{hid}} = 64$) and report the average MAE across all clients.

Figure 2 presents the results, with each subfigure visualizing a distinct hyperparameter to highlight qualitative and quantitative trends. We observe that moderate window sizes ($w \in [24, 36]$) yield the best accuracy, as very short windows fail to capture longer-term dynamics, while very long windows introduce noise and increase optimization difficulty. Regarding the EMA smoothing weight, we find that intermediate values ($\beta \in [0.80, 0.90]$) provide the best performance by stabilizing the global structural priors while still allowing timely adaptation to newly aggregated client graphs. Overly small β makes the priors sensitive to noisy round-to-round fluctuations, whereas overly large β slows down structural updates and may hinder personalization under non-IID workload shifts.

For the contrastive objective, moderate InfoNCE weights ($\lambda_1 \in [0.2, 0.4]$) improve robustness by aligning inter- and intra-graph embeddings. Too small weights do not sufficiently regularize the latent space, while too large weights overemphasize representation alignment at the expense of regression accuracy, leading to underfitting of rapid workload fluctuations. The regulation coefficient λ_2 exhibits a similar "sweet spot" behavior: a small but nonzero value ($\lambda_2 \approx 0.01$) stabilizes federated optimization under non-IID workloads by preventing client drift, whereas excessively large λ_2 overly constrains local adaptation and degrades personalization.

The Gumbel temperature τ controls the sharpness of the adaptive edge gating in AdaGCN to generate $h_i^{\text{intra},r}$ and $h_i^{\text{inter},r}$. Lower temperatures ($0.3 \leq \tau \leq 0.5$) encourage near-binary gating decisions that prune weak edges while maintaining differentiability, resulting in slightly better MAE than very smooth gating ($\tau \geq 0.7$), which tends to retain noisy or unstable connections. Finally, the hidden dimension d_{hid} shows diminishing returns beyond 64. Larger capacities (96 or 128) introduce mild overfitting and increase optimization variance, offering only marginal improvements or even slight degradation.

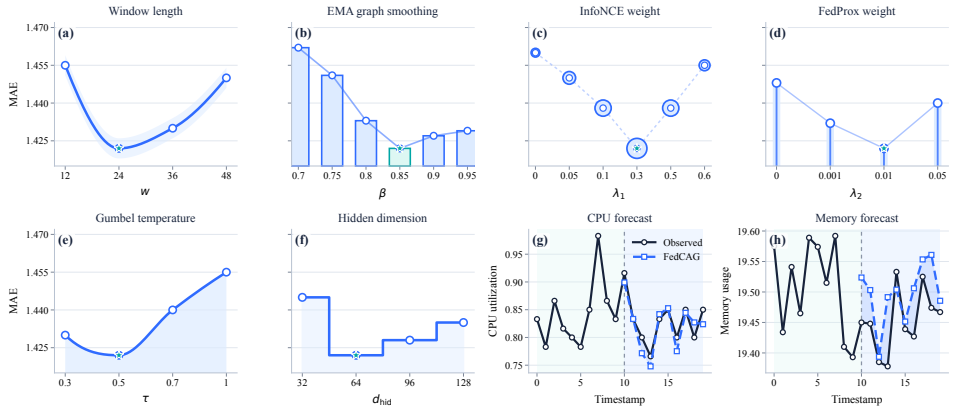


Figure 2: (a)–(f): Hyperparameter sensitivity visualization for FedCAG with respect to w, β, λ_1, λ_2, τ, and d_{hid}, respectively. (g) and (h): Prediction visualization by using FedCAG.

4.5 Prediction Analysis

Figure 2 (g) and (h) plot the predicted and actual trajectories for CPU and memory utilization in a representative client when FedCAG is utilized. These two modalities exhibit distinct temporal characteristics. CPU usage fluctuates rapidly, reflecting bursty demand patterns, whereas memory utilization evolves smoothly with relatively small variance across timesteps. For CPU utilization, the model tracks the overall shape of the ground truth and captures the local peaks that occur in the final portion of the sequence. Minor deviations occur during sharper transitions, as expected given the higher volatility and noise typically present in CPU signals. Memory utilization shows even closer alignment. Although the curve appears visually fluctuating, in absolute terms, the predicted values remain very close to the actual measurements. The results imply that FedCAG preserves the smooth temporal progression of memory usage with minimal lag or bias, and the error remains consistently low across the entire segment.

5 Conclusion

In this paper, we investigate federated multivariate workload forecasting in distributed non-IID multi-cloud environments, where strong temporal non-stationarity, heterogeneous metric dependencies, and privacy constraints make centralized prediction impractical. We propose FedCAG, a causality-aware federated dual-graph framework that develops directed inter-metric relationships, temporal structure, and lightweight personalization to enable robust collaboration across heterogeneous cloud clients without sharing raw telemetry. Experiments on real-world multi-cloud traces demonstrate that FedCAG consistently outperforms existing federated baselines and even some centralized methods, particularly under cross-provider non-IID settings and increasing system scale.

Acknowledgments

The work of Y. Luo, Z. Yang, J. Zheng, H. Wang, and D. Wu is partially supported by Hong Kong Research Grants Council grant R1018-25F.

References

[1] Dana Petcu. Multi-cloud: expectations and current approaches. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 1–6, 2013.

[2] Juncal Alonso, Leire Orue-Echevarria, Valentina Casola, Ana Isabel Torre, Maider Huarte, Eneko Osaba, and Jesus L Lobo. Understanding the challenges and novel architectural models of multi-cloud native applications–a systematic literature review. *Journal of Cloud Computing*, 12(1):6, 2023.

[3] Zhiyang Zhang, Fengli Zhang, Zehui Xiong, Kuan Zhang, and Dajiang Chen. Lsia3cs: Deep-reinforcement-learning-based cloud–edge collaborative task scheduling in large-scale iiot. *IEEE Internet of Things Journal*, 11(13):23917–23930, 2024.

[4] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. *IEEE transactions on cloud computing*, 3(4): 449–458, 2014.

[5] Menglan Hu, Hao Wang, Xiaohui Xu, Jianwen He, Yi Hu, Tianping Deng, and Kai Peng. Joint optimization of microservice deployment and routing in edge via multi-objective deep reinforcement learning. *IEEE Transactions on Network and Service Management*, 2024.

[6] Jing Bi, Haitao Yuan, Shuang Li, Kaiyi Zhang, Jia Zhang, and Mengchu Zhou. Arima-based and multiapplication workload prediction with wavelet decomposition and savitzky–golay filter in clouds. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(4):2495–2506, 2024.

[7] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[8] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

[9] Kan Guo, Yongli Hu, Yanfeng Sun, Sean Qian, Junbin Gao, and Baocai Yin. Hierarchical graph convolution network for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 151–159, 2021.

[10] He Li, Lu Yu, and Wu He. The impact of gdpr on global technology development, 2019.

[11] Lucas Franke, Huayu Liang, Sahar Farzanehpour, Aaron Brantly, James C Davis, and Chris Brown. An exploratory mixed-methods study on general data protection regulation (gdpr) compliance in open-source software. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 325–336, 2024.

[12] Fei Shen and Ruqiang Yan. A new intermediate-domain svm-based transfer model for rolling bearing rul prediction. *IEEE/ASME Transactions on Mechatronics*, 27(3):1357–1369, 2021.

[13] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282, 2017.

[14] Wenke Huang, Mang Ye, Zekun Shi, Guancheng Wan, He Li, Bo Du, and Qiang Yang. Federated learning for generalization, robustness, fairness: A survey and benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9387–9406, 2024.

[15] Danyang Xiao, BoKai Cao, and Weigang Wu. Efl-wp: Federated learning-based workload prediction in inter-cloud environments. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10, 2022.

[16] Xiaoming Yuan, Jiahui Chen, Jiayu Yang, Ning Zhang, Tingting Yang, Tao Han, and Amir Taherkordi. Fedstn: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 24(8):8738–8748, 2023.

[17] Jingxue Chen, Hang Yan, Zhiyuan Liu, Min Zhang, Hu Xiong, and Shui Yu. When federated learning meets privacy-preserving computation. *ACM Computing Surveys*, 56(12):1–36, 2024.

[18] Yudong Zhang, Xu Wang, Pengkun Wang, Binwu Wang, Zhengyang Zhou, and Yang Wang. Modeling spatio-temporal mobility across data silos via personalized federated learning. *IEEE Transactions on Mobile Computing*, 23(12):15289–15306, 2024.

[19] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 914–921, 2020.

[20] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

[21] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020.

[22] Weiyang Kong, Ziyu Guo, and Yubao Liu. Spatio-temporal pivotal graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 8627–8635, 2024.

[23] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[24] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[27] Zhengjie Yang, Wei Bao, Dong Yuan, Nguyen H Tran, and Albert Y Zomaya. Federated learning with nesterov accelerated gradient. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4863–4873, 2022.

[28] Sen Fu, Zhengjie Yang, Chuang Hu, and Wei Bao. Personalized federated learning with contrastive momentum. *IEEE Transactions on Big Data*, 2024.

[29] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[30] Rahul Dey and Fathi M. Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems*, pages 1597–1600, 2017.

[31] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

[32] Yi Liu, James J. Q. Yu, Jiawen Kang, Dusit Niyato, and Shuyu Zhang. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal*, 7(8):7751–7763, 2020.

[33] Tianqi Yu, Guodong Hua, Huaisheng Wang, Jianfeng Yang, and Jianling Hu. Federated-lstm based network intrusion detection method for intelligent connected vehicles. In *ICC 2022-IEEE International Conference on Communications*, pages 4324–4329, 2022.

[34] Chenhan Zhang, Shuyu Zhang, James J. Q. Yu, and Shui Yu. Fastgnn: A topological information protected federated learning approach for traffic speed forecasting. *IEEE Transactions on Industrial Informatics*, 17 (12):8464–8474, 2021.

[35] Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. Cross-node federated graph neural network for spatio-temporal data modeling. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 1202–1211, 2021.

[36] Hao Miao, Yan Zhao, Chenjuan Guo, Bin Yang, Kai Zheng, and Christian S. Jensen. Spatio-temporal prediction on streaming data: A unified federated continuous learning framework. *IEEE Transactions on Knowledge and Data Engineering*, 37(4):2126–2140, 2025.

[37] Hamid Mehdi, Zahra Pooranian, and Paola G Vinueza Naranjo. Cloud traffic prediction based on fuzzy arima model with low dependence on historical data. *Transactions on Emerging Telecommunications Technologies*, 33(3):e3731, 2022.

[38] Adel Abusitta, Martine Bellaiche, and Michel Dagenais. An svm-based framework for detecting dos attacks in virtualized clouds under changing environment. *Journal of Cloud Computing*, 7(1):1–18, 2018.

[39] Fahimeh Farahnakian, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Nguyen Trung Hieu, and Hannu Tenhunen. Energy-aware vm consolidation in cloud data centers using utilization prediction model. *IEEE Transactions on Cloud Computing*, 7(2):524–536, 2019.

[40] Ning Wang and Jia-Yang Li. Efficient multi-channel thermal monitoring and temperature prediction based on improved linear regression. *IEEE Transactions on Instrumentation and Measurement*, 71:1–9, 2021.

[41] Qi Zhang and Shengli Pan. An ai-augmented kalman filter approach to monitoring network traffic matrix. *IEEE Transactions on Network Science and Engineering*, 11(3):2426–2437, 2024.

[42] Ashutosh Kumar Dubey, Abhishek Kumar, Vicente García-Díaz, Arpit Kumar Sharma, and Kishan Kanhaiya. Study and analysis of sarima and lstm in forecasting time series data. *Sustainable Energy Technologies and Assessments*, 47:101474, 2021.

[43] Zheyi Chen, Jia Hu, Geyong Min, Albert Y Zomaya, and Tarek El-Ghazawi. Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. *IEEE Transactions on Parallel and Distributed Systems*, 31(4):923–934, 2019.

[44] Zhenghua Xu, Zhoutao Yu, Hexiang Zhang, Junyang Chen, Junhua Gu, Thomas Lukasiewicz, and Victor C. M. Leung. Phacia-tcns: Short-term load forecasting using temporal convolutional networks with parallel hybrid activated convolution and input attention. *IEEE Transactions on Network Science and Engineering*, 11(1):427–438, 2024.

[45] Kai Jia, Xiao Yu, Chen Zhang, Wenhua Hu, Dongdong Zhao, and Jianwen Xiang. Software aging prediction for cloud services using a gate recurrent unit neural network model based on time series decomposition. *IEEE Transactions on Emerging Topics in Computing*, 11(3):580–593, 2023.

[46] Yao Lu, Yongjing Shang, Jie Cui, Hong Zhong, Lu Liu, and Geyong Min. Fp-mlp: A frequency domain patch-based mlp model for gpu-dominated cloud workload prediction. *IEEE Transactions on Cloud Computing*, 2025.

[47] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[48] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[49] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286, 2022.

[50] Haitao Yuan, Qinglong Hu, Meijia Wang, Shen Wang, Jing Bi, Rajkumar Buyya, Shuyuan Shi, Jinhong Yang, Jia Zhang, and MengChu Zhou. Data-filtered prediction with decomposition and amplitude-aware permutation entropy for workload and resource utilization in cloud data centers. *IEEE Internet of Things Journal*, 2025.

[51] Siyu Cheng, Dong Jin, Yuanyi Ma, Shuangwu Chen, Huasen He, and Jian Yang. Linkguard: Link flooding attack detection and mitigation via spatio-temporal graph convolutional network. *IEEE Transactions on Network Science and Engineering*, 11(5):4059–4075, 2024.

[52] Yongcan Luo, Jingxuan Chen, Shuxin Qin, and Dapeng Wu. Dual graph learning for multivariate time series anomaly detection in iout. *IEEE Transactions on Network Science and Engineering*, 2025.

[53] Yongcan Luo, Jiahao Zheng, Zhengjie Yang, Ning Chen, and Dapeng Wu. Pleno-alignment framework for stock trend prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 36(9):16604–16618, 2025.

[54] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[55] Yang Luo, Mohan Gao, Zhemeng Yu, Haoyuan Ge, Xiaofeng Gao, Tengwei Cai, and Guihai Chen. Integrating system state into spatio temporal graph neural network for microservice workload prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5521–5531, 2024.

[56] Huanfei Ma, Kazuyuki Aihara, and Luonan Chen. Detecting causality from nonlinear dynamics with short-term time series. *Scientific reports*, 4(1):7464, 2014.

[57] Chainarong Amornbunchornvej, Elena Zheleva, and Tanya Berger-Wolf. Variable-lag granger causality and transfer entropy for time series analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(4):1–30, 2021.

[58] William Todo, Merwann Selmani, Béatrice Laurent, and Jean-Michel Loubes. Counterfactual explanation for multivariate times series using a contrastive variational autoencoder. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[59] Edo Cohen-Karlik, Avichai Ben David, and Amir Globerson. Regularizing towards permutation invariance in recurrent models. *Advances in Neural Information Processing Systems*, 33:18364–18374, 2020.

[60] Chuanting Zhang, Shuping Dang, Basem Shihada, and Mohamed-Slim Alouini. Dual attention-based federated learning for wireless traffic prediction. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–10, 2021.

[61] Yuris Mulya Saputra, Dinh Thai Hoang, Diep N. Nguyen, Eryk Dutkiewicz, Markus Dominik Mueck, and Srikathyayani Srikanteswara. Energy demand prediction with federated learning for electric vehicle networks. In *2019 IEEE Global Communications Conference*, pages 1–6, 2019.

[62] Mahmoud M. Badr, Mohamed M. E. A. Mahmoud, Yuguang Fang, Mohammed Abdulaal, Abdulah Jeza Aljohani, Waleed Alasmary, and Mohamed I. Ibrahem. Privacy-preserving and communication-efficient energy prediction scheme based on federated learning for smart grids. *IEEE Internet of Things Journal*, 10 (9):7719–7736, 2023.

[63] Pouya M Ghari and Yanning Shen. Personalized federated learning with mixture of models for adaptive prediction and model fine-tuning. *Advances in Neural Information Processing Systems*, 37:92155–92183, 2024.

[64] Xinxin Feng, Haoran Sun, Shunjian Liu, Junxin Guo, and Haifeng Zheng. Federated meta-learning on graph for traffic flow prediction. *IEEE Transactions on Vehicular Technology*, 73(12):19526–19538, 2024.

[65] Liuyi Yao, Zhen Wang, Yuexiang Xie, Yaliang Li, Weirui Kuang, Daoyuan Chen, and Bolin Ding. Is sharing neighbor generator in federated graph learning safe? *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8568–8579, 2024.

[66] Dimitrios Brodimas, Dimitrios Kapolos, Eleftherios Mylonas, Alexios Birbas, and Michael Birbas. Federated learning for workload forecasting for network service management. In *2025 14th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4, 2025.

[67] Lumin Liu, Jun Zhang, S.H. Song, and Khaled B. Letaief. Client-edge-cloud hierarchical federated learning. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[68] Yangguang Cui, Kun Cao, Junlong Zhou, and Tongquan Wei. Optimizing training efficiency and cost of hierarchical federated learning in heterogeneous mobile-edge cloud computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(5):1518–1531, 2023.

[69] Shinu M Rajagopal, M Supriya, and Rajkumar Buyya. Fedsdm: Federated learning based smart decision making module for ecg data in iot integrated edge–fog–cloud computing environments. *Internet of Things*, 22:100784, 2023.

[70] Syed Thouheed Ahmed, V. Vinoth Kumar, and Junho Jeong. Heterogeneous workload-based consumer resource recommendation model for smart cities: ehealth edge–cloud connectivity using federated split learning. *IEEE Transactions on Consumer Electronics*, 70(1):4187–4196, 2024.

[71] Chuanting Zhang, Haixia Zhang, Shuping Dang, Basem Shihada, and Mohamed-Slim Alouini. Gradient compression and correlation driven federated learning for wireless traffic prediction. *IEEE Transactions on Cognitive Communications and Networking*, 11(4):2246–2258, 2025.

[72] Hui Ma, Kai Yang, and Yang Jiao. Cellular traffic prediction via byzantine-robust asynchronous federated learning. *IEEE Transactions on Network Science and Engineering*, 12(4):2402–2414, 2025.

[73] M. Saeid HaghighiFard and Sinem Coleri. Hierarchical federated learning in multi-hop cluster-based vanets. *IEEE Transactions on Vehicular Technology*, 74(10):15371–15385, 2025.

[74] Jun Du, Bingqing Jiang, Chunxiao Jiang, Yuanming Shi, and Zhu Han. Gradient and channel aware dynamic scheduling for over-the-air computation in federated edge learning systems. *IEEE Journal on Selected Areas in Communications*, 41(4):1035–1050, 2023.

Appendices Contents

A Related Work **16**

A.1 Centralized Time-series Prediction 16

A.2 Federated Time-series and Spatio-temporal Forecasting 16

A.3 Federated Learning at Large and Cloud Applications 17

B Graph Generation **17**

B.1 Causality Graph Generation 17

B.2 Spatial-temporal Dual Graphs 18

B.3 Dual-graph Message Passing 19

C Details on Local Optimization **19**

D Training and Inference of FedCAG **20**

E Dataset and MUCEP **21**

F Implementation Details **22**

G Baseline Details **22**

H Client Dropout Robustness **23**

I Personalization and Head Decomposition Study **23**

J Inference Time **24**

K Additional Experimental Results **25**

L Mechanism-Level Ablation **25**

M Training-Time Breakdown **26**

N Convergence Analysis **26**

O Limitations and Discussion **28**

P Broader Impact **29**

A Related Work

Workload forecasting has long been a core problem in large-scale distributed computing systems, as accurate prediction directly underpins resource allocation, autoscaling, and service-level management. Existing research spans a broad range, from early centralized statistical models to modern deep learning architectures, and more recently to privacy-preserving federated learning frameworks. This section reviews prior work along three dimensions: centralized time-series prediction for cloud workloads, federated learning approaches for time-series and spatio-temporal forecasting, and federated learning at large from a systems and application perspective. We highlight both the strengths and limitations of these approaches, particularly under non-IID and multi-cloud settings, which motivate the design choices of our proposed framework.

A.1 Centralized Time-series Prediction

Early research on workload prediction in large-scale computing systems primarily relies on centralized statistical and machine-learning models, assuming that all telemetry data are available at a single location. Classical time-series approaches such as ARIMA [37, 4], Support Vector Machines (SVMs) [38, 12], and regression-based models [39, 40] are widely adopted due to their simplicity, low computational overhead, and interpretability. Extensions including SARIMA, Kalman filtering [41], and exponential smoothing improve robustness to noise, seasonality, and missing data [42]. Despite these advantages, such methods rely heavily on stationarity assumptions and limited dependency structures, making them ill-suited for modern cloud environments characterized by autoscaling, dynamic reconfiguration, and bursty cross-service interactions [5].

With the increasing availability of high-frequency and multivariate telemetry, Deep Learning (DL) has become the dominant paradigm for centralized workload forecasting. Recurrent architectures such as RNNs [43] and LSTMs [31] capture temporal dependencies but often struggle with long horizons and volatile traces. Temporal convolutional networks and MLP-based forecasters offer improved stability and efficiency through parallelism and receptive-field expansion [44–46]. Attention-based models further enhance long-range modeling capabilities, as exemplified by Autoformer [7], Informer [47], Reformer [48], and Fedformer [49]. These methods exploit decomposition, frequency-domain operations, and patching strategies to scale to long sequences [50].

Beyond purely temporal modeling, spatio-temporal graph neural networks (ST-GNNs) explicitly encode structural dependencies among metrics, services, or infrastructure components [9, 19, 51–53]. Representative approaches such as DCRNN [54], Graph WaveNet [20], and MTGNN [21] combine graph convolutions with temporal dynamics to model load propagation. More recent works incorporate adaptive or learned adjacency and domain-specific structures for microservices and cloud systems [55, 22]. However, most centralized ST-GNNs rely on correlation-based or symmetric adjacency, which can blur directionality and limit robustness under interventions and distribution shifts [56]. Causality-inspired extensions based on Granger causality or transfer entropy have been proposed to address these limitations [57–59], yet they still assume centralized access to all telemetry data.

Although centralized models currently provide strong predictive performance, their reliance on raw data aggregation poses significant challenges in practice. Privacy constraints, organizational boundaries, and inter-cloud governance increasingly limit the feasibility of centralized training, motivating alternative learning paradigms.

A.2 Federated Time-series and Spatio-temporal Forecasting

Federated Learning (FL) has emerged as a natural solution for time-series prediction in settings where data sharing is restricted. In the standard federated paradigm, clients train local predictors on private sequences and periodically share model updates with a coordinator, enabling collaborative learning without exposing raw data. Early studies apply this framework to traffic forecasting [60] and energy demand prediction [61, 62], demonstrating that federated predictors can approach centralized accuracy while significantly reducing data transfer.

A key challenge in federated time-series forecasting is the statistical heterogeneity across clients. Workloads observed by different sensors, regions, or organizations often follow distinct temporal patterns. To mitigate this issue, several works introduce clustering, personalized aggregation, or

attention-based weighting to replace classic FedAvg [60, 61]. Continual and online federated forecasting further addresses non-stationarity by enabling models to adapt to evolving data streams while preserving historical knowledge [63, 36]. These studies highlight that personalization is essential for maintaining accuracy under non-IID temporal dynamics.

Many real-world time-series problems are inherently relational, motivating federated extensions of spatio-temporal graph learning. Federated graph learning integrates Graph Neural Networks (GNNs) with FL to capture structured dependencies while preserving privacy [32]. Applications include traffic flow prediction, where spatial topology and semantic relations are combined with temporal signals under secure aggregation [16]. Meta-learning and transferable initialization have also been explored to adapt graph models across heterogeneous topologies [64]. However, federated graph settings introduce additional privacy risks, as shared embeddings or structural summaries may leak sensitive relational information [65]. These challenges suggest that effective federated spatio-temporal forecasting must jointly address non-IID dynamics, heterogeneous graph structures, and privacy-aware structural sharing.

Within cloud and inter-cloud environments, federated workload forecasting is particularly relevant. Centralized training on fine-grained traces raises privacy and ownership concerns, especially when workloads span multiple providers. EFL-WP explicitly targets inter-cloud workload prediction and addresses non-IID traces via participant selection and multi-global aggregation [15]. Related efforts explore federated load forecasting for MANO frameworks [66], hierarchical edge–cloud aggregation [67, 68], and vertical domains such as eHealth [69, 70]. These studies demonstrate the promise of FL for cloud-centric forecasting, while also revealing open challenges in scalability, strong cross-provider heterogeneity, and structural modeling under privacy constraints.

A.3 Federated Learning at Large and Cloud Applications

Beyond time-series forecasting, FL has been extensively studied as a general paradigm for privacy-preserving and distributed intelligence across domains. System-level research focuses on improving communication efficiency through sparsification, compression, and correlation-aware aggregation [71]. Secure aggregation and encryption protocols strengthen privacy guaranties while maintaining convergence [62]. Robust FL addresses unreliable or adversarial clients via differential privacy and distributionally robust optimization [72].

In highly dynamic environments such as vehicular and wireless networks, hierarchical FL and cluster-based coordination improve robustness to intermittent connectivity and heterogeneous data distributions [73]. Over-the-air computation and client scheduling further optimize participation under bandwidth and channel constraints [74]. These advances underscore that modern FL systems must balance accuracy, efficiency, robustness, and privacy.

Overall, existing works establish FL as a powerful alternative to centralized training, yet applying it to large-scale, heterogeneous, and relational time-series systems remains challenging. These challenges motivate federated architectures that explicitly model structural dependencies, account for causality and directionality, and support lightweight personalization, which the proposed FedCAG framework addresses.

B Graph Generation

B.1 Causality Graph Generation

To capture directional influence between metrics, we construct a causality matrix $\mathbf{C}_i^r \in \mathbb{R}^{n\times n}$ for each client i at communication round r, where the entry $C_{i,u\leftarrow v}^r$ quantifies the directional effect from metric v to metric u ($u \neq v$). Motivated by transfer entropy, we adopt a simplified information-theoretic proxy that compares the current value of metric u with the historical mean of metric v over the previous w steps. Let $X_i^t \in \mathbb{R}^n$ denote the multivariate telemetry at time t on client i, and $X_{i,u}^t$ be its u-th metric. For each ordered pair (u, v) with $u \neq v$, we define

$$\bar{X}_{i,v}^{\,t-w:t-1} = \frac{1}{w}\sum_{\tau=t-w}^{t-1} X_{i,v}^{\tau}, \quad \tilde{X}_{i,u}^{t} = \max\big(X_{i,u}^{t}, \varepsilon\big), \tag{13}$$

where $\varepsilon > 0$ (e.g., 10^{-6}) ensures numerical stability. The directional score from v to u is approximated as

$$C^r_{i,u\leftarrow v} = \frac{1}{t_0 - w} \sum_{t=w+1}^{t_0} \tilde{X}^t_{i,u} \log \frac{\tilde{X}^t_{i,u}}{\max\big(\bar{X}_{i,v}^{\,t-w:t-1}, \varepsilon\big)}, \tag{14}$$

where t_0 is the most recent timestamp available on client i at round r. Intuitively, $C^r_{i,u\leftarrow v}$ becomes large when high recent values of metric v are consistently associated with larger current values of metric u, indicating a strong directional influence. We set $C^r_{i,u\leftarrow u} = 0$ to avoid trivial self-causality.

To obtain a sparse directed causality graph, we threshold $\mathbf{C}^r_i$ at a client-specific percentile. Let $q_\kappa(\mathbf{C}^r_i)$ denote the κ-th percentile of the off-diagonal entries of $\mathbf{C}^r_i$, and define the causality edge set and weights as

$$\mathcal{E}^{(i)}_{\text{cau}} = \big\{(v,u) \;:\; C^r_{i,u\leftarrow v} > q_\kappa(\mathbf{C}^r_i)\big\}, \quad w^r_{i,u\leftarrow v} = C^r_{i,u\leftarrow v}. \tag{15}$$

In practice, $\mathcal{E}^{(i)}_{\text{cau}}$ and $w^r_{i,u\leftarrow v}$ correspond to the sparse edge index and edge attribution tensors fed into the GNN layers.

Why Transfer Entropy (TE)? Eq. (14) is a causality-inspired and communication-efficient proxy designed to extract a directed predictive dependency graph under federated constraints, rather than a full nonparametric estimator of TE. Recall that TE from metric v to metric u on client i can be written as the conditional information gain

$$\text{TE}^{(i)}_{v\to u} = \mathbb{E}\left[\log \frac{p\big(X^t_{i,u} \mid X^{t-1:t-w}_{i,u},\, X^{t-1:t-w}_{i,v}\big)}{p\big(X^t_{i,u} \mid X^{t-1:t-w}_{i,u}\big)}\right], \tag{16}$$

which measures how much the past of v reduces the uncertainty of $X^t_{i,u}$ beyond the past of u. Directly estimating these conditional densities is expensive and statistically unstable in high-dimensional multivariate telemetry, and is further complicated in federated settings where raw sequences cannot be centralized. FedCAG therefore adopts a lightweight approximation that isolates the contribution of v's recent history by summarizing $X^{t-w:t-1}_{i,v}$ via $\bar{X}_{i,v}^{\,t-w:t-1}$, while the target history $X^{t-1:t-w}_{i,u}$ is modeled separately by the temporal encoder and the temporal graph $\mathbf{A}^r_i$. In this sense, our graph construction is a structural prior complementary to the temporal modeling branch rather than a standalone causal estimator.

More concretely, consider a mean-field approximation where the conditional effect of v on $X^t_{i,u}$ is represented by a positive predictor $p(X^t_{i,u} \mid X^{t-1:t-w}_{i,u}, X^{t-1:t-w}_{i,v}) \approx q(X^t_{i,u} \mid \bar{X}_{i,v}^{\,t-w:t-1})$. Under common exponential-family observation models for nonnegative telemetry (e.g., Poisson/Gamma intensities), the log-likelihood ratio term in $\text{TE}^{(i)}_{v\to u}$ contains a component proportional to $X^t_{i,u} \log \frac{X^t_{i,u}}{\hat{X}^t_{i,u}(v)}$ up to constants, where $\hat{X}^t_{i,u}(v)$ is a function of v's recent history. Eq. (14) instantiates this idea with $\hat{X}^t_{i,u}(v) = \max(\bar{X}_{i,v}^{\,t-w:t-1}, \varepsilon)$ and $\tilde{X}^t_{i,u}$ as a stability-aware importance weight, yielding a directed score that increases when large values of $X^t_{i,u}$ systematically coincide with elevated recent values of v. Importantly, the construction is inherently *asymmetric* because it compares the current target $X^t_{i,u}$ against the *past* summary of $X^{t-w:t-1}_{i,v}$, and swapping (u, v) generally produces a different score, which provides a directed structural signal beyond symmetric correlation.

Finally, while Eq. (14) does not claim the identifiability of true causal effects, it serves the intended role in FedCAG: producing a directed, sparse dependency graph that improves forecasting by guiding inter-metric message passing and capturing lead-lag patterns that vary across clients.

B.2 Spatial-temporal Dual Graphs

From a multivariate time-series perspective, each client i must simultaneously capture dependencies across metrics (e.g., CPU, memory, and network I/O on the same VM) and along time (e.g., bursts and local trends). To this end, we construct two complementary graphs: a temporal intra-window graph $\mathbf{A}^r_i$ and a spatial inter-metric graph $\mathbf{B}^r_i$.

The temporal graph $\mathbf{A}^r_i \in \mathbb{R}^{w\times w}$ models temporal dependencies within a window of w time steps. Each node corresponds to a time index inside the sliding window, and edges encode short-range

temporal interactions. We adopt a banded and locally connected structure in which non-zero entries are concentrated around the main diagonal, capturing that each time step is most strongly related to its recent neighbors. This chain-like topology provides a simple temporal prior that is later refined by the adaptive edge-selection mechanism.

The spatial graph $\mathbf{B}_i^r \in \mathbb{R}^{n\times n}$ describes how metrics interact at client i. Each node corresponds to a metric, and edges reflect cross-metric dependencies arising from shared resources, co-located services, or correlated workload patterns. We initialize $\mathbf{B}_i^r$ directly from the sparsified causality structure in (15), assigning non-zero entries to metric pairs with strong directional influence (large $C^{(i)}_{u\leftarrow v}$). Thus, $\mathbf{C}_i^r$ serves both as an explicit directed prior and as the backbone for the spatial adjacency, while the subsequent adaptive graph convolution further refines which edges are effectively used during message passing.

B.3 Dual-graph Message Passing

To jointly exploit spatial and temporal structures, each client i forms two graph-structured inputs from a window $X^{(i)}_{t_1:t_w} \in \mathbb{R}^{w\times n}$: (i) an intra-temporal graph based on $\mathbf{A}_i^r$ and (ii) an inter-metric graph based on $(\mathbf{B}_i^r, \mathbf{C}_i^r)$. These graphs are processed in parallel by dedicated GNN branches. Following the general GNN formulation, for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node features $\{\mathbf{h}_v^{k-1}\}_{v\in\mathcal{V}}$ and edge features $\{\mathbf{j}_{uv}\}_{(u,v)\in\mathcal{E}}$, the k-th layer performs aggregation and update:

$$\begin{aligned} \mathbf{m}^k_{\mathcal{N}_v} &= \mathrm{AGG}\big(\{(\mathbf{h}_v^{k-1}, \mathbf{h}_u^{k-1}, \mathbf{j}_{uv}) \mid u \in \mathcal{N}_v\}\big), \\ \mathbf{h}_v^k &= \mathrm{UPDATE}(\mathbf{h}_v^{k-1}, \mathbf{m}^k_{\mathcal{N}_v}), \end{aligned} \tag{17}$$

where $\mathcal{N}_v$ denotes the neighbor set of node v and $\mathbf{m}^k_{\mathcal{N}_v}$ is the aggregated message. $\mathrm{AGG}(\cdot)$ is a permutation-invariant neighborhood aggregation operator, and $\mathrm{UPDATE}(\cdot)$ maps the previous node state and the aggregated message to the next-layer representation.

In FedCAG, the inter-metric and intra-temporal graphs are fed into separate AdaGCN layers followed by GAT layers. For each candidate edge (u, v), AdaGCN maintains a learnable logit vector $\boldsymbol{\ell}_{uv}$ and applies a Gumbel Softmax gate:

$$\mathbf{z}_{uv} = \mathrm{Softmax}\left(\frac{\boldsymbol{\ell}_{uv} + \boldsymbol{\epsilon}_{uv}}{\tau}\right), \quad \gamma_{uv} = z_{uv,0}, \tag{18}$$

where $\boldsymbol{\epsilon}_{uv}$ is a Gumbel noise vector and τ is a temperature parameter. The scalar $\gamma_{uv} \in [0, 1]$ serves as a differentiable, sample-based indicator of whether the edge (u, v) is active. Given edge weights ω_{uv}, the message from node u to node v becomes

$$\mathbf{m}_{u\to v} = \gamma_{uv}\, \omega_{uv}\, \mathbf{h}_u^{k-1}\mathbf{W}, \tag{19}$$

where $\mathbf{W}$ is a learnable weight matrix. This mechanism allows the model to start from structured priors $(\mathbf{A}_i^r, \mathbf{B}_i^r, \mathbf{C}_i^r)$ while adaptively pruning or reweighting edges in a data-driven manner.

Applying this dual-graph encoder yields the inter- and intra-embeddings used in (20):

$$\begin{aligned} h_i^{\mathrm{intra},r} &= \mathrm{GNN}(\mathbf{A}_i^r), \\ h_i^{\mathrm{inter},r} &= \mathrm{GNN}(\mathbf{B}_i^r, \mathbf{C}_i^r), \end{aligned} \tag{20}$$

which are further fused through causality-aware multi-head attention in (4). In this way, each client i jointly leverages a directed causality graph, a causality-informed spatial graph over metrics, and a temporal graph over window indices to capture rich spatial–temporal dependencies in multi-cloud workload traces.

C Details on Local Optimization

First, the regression term

$$\mathcal{L}_{\mathrm{MSE}} = \|\hat{X}_i^{t+1} - X_i^{t+1}\|_2^2 \tag{21}$$

encourages accurate prediction of all n metrics in the next time step. Second, to ensure that the spatial (inter-metric) embedding $h_i^{\mathrm{inter},r}$ and the temporal (intra-window) embedding $h_i^{\mathrm{intra},r}$ represent

Algorithm 1 Training of FedCAG

Input: M, R, E
Output: $\Theta^R, \mathbf{A}^R, \mathbf{B}^R, \mathbf{C}^R$

1: Initialize global model parameters Θ^0 and global graphs $\mathbf{A}^0, \mathbf{B}^0, \mathbf{C}^0$. Initialize Ψ_i^0 for all clients.
2: **for** $r = 1$ **to** R **do**
3: **for all** clients $i \in [M]$ in parallel **do**
4: Receives $\Theta^{r-1}, \mathbf{A}^{r-1}, \mathbf{B}^{r-1}, \mathbf{C}^{r-1}$.
5: Construct & personalize local graphs $\hat{\mathbf{A}}_i^r, \hat{\mathbf{B}}_i^r, \hat{\mathbf{C}}_i^r$.
6: **for** $e = 1$ **to** E **do**
7: Compute $h_i^{\text{intra},r,e}, h_i^{\text{inter},r,e}$ and fuse $h_i^{\text{fuse},r,e}$.
8: Update $\Theta_i^{r,e}$ and $\Psi_i^{r,e}$ by minimizing (6).
9: **end for**
10: Upload $\Theta_i^{r,E}, \mathbf{A}_i^{r,E}, \mathbf{B}_i^{r,E}, \mathbf{C}_i^{r,E}$ to server.
11: **end for**
12: For the server:
13: Aggregate global graphs $\mathbf{A}^r, \mathbf{B}^r, \mathbf{C}^r$ via (7) and (8).
14: Aggregate global model parameters Θ^r via (9).
15: Broadcast $\Theta^r, \mathbf{A}^r, \mathbf{B}^r, \mathbf{C}^r$ to clients.
16: **end for**

consistent workload states, we employ an InfoNCE contrastive loss. For each anchor $h_i^{\text{inter},r}$, we treat its paired temporal embedding $h_i^{\text{intra},r}$ as a positive and embeddings from other windows within the same training batch as negatives. The loss is defined as

$$\mathcal{L}_{\text{Info}} = -\log \frac{\exp\Big(\text{sim}(h_i^{\text{inter},r}, h_i^{\text{intra},r})/\tau\Big)}{\sum_{h^- \in \mathcal{N}^{(i)}} \exp\Big(\text{sim}(h_i^{\text{inter},r}, h^-)/\tau\Big)}, \tag{22}$$

where $\text{sim}(\cdot,\cdot)$ is cosine similarity, τ is a temperature parameter, and $\mathcal{N}^{(i)}$ includes all negatives from the same training batch. The contrastive alignment encourages the two graph-based encoders to produce semantically coherent latent representations, which improves robustness under noisy, bursty, or partially missing workloads common across multi-cloud telemetry streams. Finally, the regulation term (inspired by FedProx [26])

$$\lambda_2 \left\| \Theta_i^r - \Theta^{r-1} \right\|_2^2 \tag{23}$$

regularizes local updates against the global parameters, which stabilizes training when client workloads are non-IID, preventing local models from drifting away from the global model's update direction/path.

D Training and Inference of FedCAG

The training process of FedCAG is illustrated in Algorithm 1. The FedCAG algorithm is implemented with R communication rounds, with each round containing E local epochs. At the beginning of each communication round $\forall r \in [1, R]$, all clients receive the last round global model parameters and global graphs (Line 4 in Algorithm 1), and construct and personalize local graphs (Line 5). Then, clients enter the local training process $\forall e \in [1, E]$ to train local models as well as private adapters, and update local graphs (Lines 6–9) for E local epochs. Afterwards, clients upload graphs and models to the server for aggregation (Line 10). Finally, the server aggregates local graphs and models, and distributes the aggregated global graphs and models back to each client for the next round of training (Lines 12–15).

Algorithm 2 summarizes the client-side inference procedure of FedCAG. After completing the training process, each client i receives the final global model parameters Θ^R (including the aggregated prediction head $\Theta^{\text{head},R}$) from the server, while keeping its private adapter Ψ_i^R locally. Each client also computes and fixes its personalized structural priors/graphs $(\hat{\mathbf{A}}_i^R, \hat{\mathbf{B}}_i^R, \hat{\mathbf{C}}_i^R)$, which encode intra-window temporal adjacency, undirected inter-metric interactions, and directed lead-lag influence

Algorithm 2 Client-side Inference of FedCAG

Require: Global model parameters Θ^R, private adapter Ψ_i^R, and global graphs $(\mathbf{A}^R, \mathbf{B}^R, \mathbf{C}^R), \forall i \in [M]$

Ensure: Predictions $\hat{X}_i^{t+1}$ and evaluation metrics on client i

1: Load Θ^r into the shared backbone and global prediction head.
2: Each client i computes and fix personalized graphs $(\hat{\mathbf{A}}_i^R, \hat{\mathbf{B}}_i^R, \hat{\mathbf{C}}_i^R)$.
3: **for** $t - w + 1$ **to** t **do**
4: Form window input $X_i^{t-w+1:t} = \{X_i^{t-w+1}, \ldots, X_i^t\}$
5: Compute intra-temporal embedding h_i^{intra} using $(X_i^{t-w+1:t}, \hat{\mathbf{A}}_i^R)$.
6: Compute inter-metric embedding h_i^{inter} using $(X_i^{t-w+1:t}, \hat{\mathbf{B}}_i^R, \hat{\mathbf{C}}_i^R)$.
7: Derive causality descriptor $c_i = \phi(\hat{\mathbf{C}}_i^R)$.
8: Fuse representations to obtain h_i^{fuse} via (4).
9: Predict next-step workload $\hat{X}_i^{t+1}$ via (5).
10: **end for**
11: Compute client i's MAE and RMSE over $\hat{X}_i^{t+1}$.
12: **Return** $\hat{X}_i^{t+1}$ and metrics MAE, RMSE for each client i.

across metrics, respectively. These components jointly define a privacy-preserving personalized predictor that can be executed fully on-device without exposing raw telemetry.

For all time steps in $[t - w + 1, t]$, each client i forms a windowed input $X_i^{t-w+1:t}$ and computes two complementary representations: an intra-temporal embedding h_i^{intra} guided by $\hat{\mathbf{A}}_i^R$, and an inter-metric embedding h_i^{inter} guided by $\hat{\mathbf{B}}_i^R$, and $\hat{\mathbf{C}}_i^R$. A compact causality descriptor $c_i^r = \phi(\hat{\mathbf{C}}_i^R)$ is further derived to summarize the dominant directional dependencies. FedCAG then applies causality-aware multi-head attention to fuse these signals into h_i^{fuse}, and outputs the next-step workload prediction $\hat{X}_i^{t+1}$ via the prediction head together with the client-private adapter. Finally, the client aggregates predictions across the test horizon and evaluates forecasting quality using standard metrics such as MAE and RMSE.

E Dataset and MUCEP

Dataset. Our workload forecasting dataset is collected from a set of heterogeneous Virtual Machines (VMs) running on Tencent Cloud and Alibaba Cloud and managed by the Purple Mountain Laboratories MUCEP platform. The monitored instances span multiple regions and availability zones, and cover diverse VM families (e.g., general-purpose, memory-optimized, and GPU-enabled nodes), leading to highly non-IID telemetry patterns across clients. For each VM, MUCEP continuously records multivariate system metrics (e.g., CPU utilization, memory usage, and network I/O) with unified interfaces across cloud vendors. The details of the VMs are listed in Table 3.

MUCEP Platform. MUCEP serves as the underlying multi-cloud control and monitoring substrate, enabling unified management of geographically distributed VMs without requiring centralized access to raw telemetry. Unlike conventional single-cloud monitoring pipelines, MUCEP provides an integrated cloud-network orchestration layer that supports cross-cloud resource visibility, traffic engineering, and service-level isolation. Its architecture separates the control logic from the forwarding layer, allowing MUCEP to abstract heterogeneous cloud networking primitives and enforce consistent connectivity policies across Tencent and Alibaba deployments. In addition, MUCEP supports multiple secure interconnection mechanisms (e.g., SD-WAN and tunneling-based overlays) and exposes fine-grained traffic statistics and resource states for downstream learning tasks. As a result, MUCEP enables us to evaluate workload forecasting under realistic multi-cloud conditions where both system capacity and network behavior vary significantly across clients, while maintaining strict locality of each site's data.

Table 3: Configurations of Virtual Machines Used in Experiments.

Cloud	Region	Type	CPU	Memory	Bandwidth
Tencent	Shanghai Zone 4	GN7 GPU	8 cores	32GB	200Mbps
Tencent	Shanghai Zone 5	S4 Standard	4 cores	8GB	1Mbps
Tencent	Beijing Zone 6	S6 Standard	2 cores	4GB	50Mbps
Tencent	Nanjing Zone 1	S6 Standard	2 cores	8GB	100Mbps
Tencent	Nanjing Zone 1	MA3 Memory-Optimized	8 cores	64GB	200Mbps
Alibaba	Beijing Zone H	ecs.s6-c1m2.xlarge	4 vCPU	8GB	100Mbps
Alibaba	Beijing Zone H	ecs.s6-c1m2.xlarge	4 vCPU	8GB	100Mbps
Alibaba	Beijing Zone H	ecs.s6-c1m2.xlarge	4 vCPU	8GB	100Mbps
Alibaba	Beijing Zone H	ecs.s6-c1m2.xlarge	4 vCPU	8GB	100Mbps
Alibaba	Zhangjiakou Zone A	ecs.t6-c1m4.xlarge	4 vCPU	16GB	200Mbps
Alibaba	Nanjing East 5	ecs.s6-c1m4.large	2 vCPU	8GB	50Mbps
Alibaba	Nanjing East 5	ecs.t6-c1m2.large	2 vCPU	4GB	50Mbps
Alibaba	Hangzhou Zone I	ecs.c5.2xlarge	8 vCPU	16GB	200Mbps
Alibaba	Hangzhou Zone I	ecs.c5.2xlarge	8 vCPU	16GB	50Mbps
Alibaba	Hangzhou Zone I	ecs.c7.xlarge	4 vCPU	8GB	50Mbps

F Implementation Details

All experiments are implemented in Python using PyTorch (v2.0.1) and PyTorch Geometric with an NVIDIA A100 GPU. Each VM trace is standardized via z-score normalization and split into training and test sets at an 80/20 ratio, using a fixed random seed (42). Models are trained with mini-batches of size 16. We train the model for $R = 20$ federated communication rounds, with each client participating in the aggregation during each training round. During each round, each client performs $E = 10$ local training epochs on its private data before sending updates to the server.

For FedCAG, the window length is set to $w = 24$ with one-step-ahead prediction. The dual-graph encoder uses a hidden dimension of 64 and four attention heads in both inter- and intra-graph GAT layers. Client-side causality matrices $\mathbf{C}_i^r$ are computed using the same lookback length and sparsified by retaining the top 20% off-diagonal entries (κ=80), forming directed inter-metric graphs. Temporal graphs follow a bidirectional chain structure within each window. Adaptive edge selection is enabled through AdaGCN with straight-through Gumbel-Softmax gates ($\tau = 0.5$). Models are optimized using AdamW with learning rate $\eta = 10^{-3}$, while the final objective employs $\lambda_1 = 0.3$ and $\lambda_2 = 0.01$ to stabilize training under non-IID workloads. All other hyperparameters remain fixed unless explicitly varied in sensitivity studies.

G Baseline Details

Considering the workload prediction and time-series analysis tasks, the baselines are categorized into two groups to provide different perspectives for comparison with our FedCAG. Centralized methods are trained on all available data jointly without considering privacy constraints, whereas federated methods operate under data isolation and communication limitations and therefore better reflect real-world deployment settings. To ensure a fair comparison, all experiments are conducted on the same computational platform and report the average results across 10 runs, with performance measured across all metrics.

Centralized baselines.

- **ARIMA** [29]: A classical linear forecasting model that achieves stationarity through differencing and fits autoregressive, moving-average, and seasonal components to capture short-term dynamics.
- **GRU** [30]: A gated recurrent unit that models temporal dependencies through update and reset gates, offering strong performance with a lower parameter footprint than LSTM.
- **LSTM** [31]: A recurrent architecture with gated memory cells designed to capture long-range temporal dependencies while mitigating vanishing-gradient issues.

- **Autoformer** [7]: A Transformer-based long-horizon forecaster that decomposes time-series components and replaces self-attention with auto-correlation modules to better capture periodicity.
- **GPT4TS** [8]: A pretrained Transformer for time series that freezes backbone layers and fine-tunes lightweight task-specific heads, leveraging general-purpose sequence representations.
- **STSGCN** [19]: A spatial-temporal synchronous graph convolutional network that jointly extracts local spatial and temporal correlations through synchronized graph filters.
- **HGCN** [9]: A hierarchical GCN that integrates fine-grained node interactions with coarse regional structures to capture multi-scale spatial relationships.
- **STPGNN** [22]: A spatial-temporal graph neural network that identifies influential nodes and applies distinct propagation mechanisms to pivotal and non-pivotal nodes, improving modeling of heterogeneous dependencies.

Federated baselines.

- **FedGRU** [32]: A federated GRU forecaster that performs local recurrent modeling and aggregates model parameters using FedAvg, with extensions for client clustering and communication efficiency.
- **FL-LSTM** [33]: A federated LSTM framework for in-vehicle intrusion detection, where vehicles locally train sequence models on CAN data and a server aggregates parameters without accessing raw signals.
- **FASTGNN** [34]: A federated spatio-temporal GNN for traffic forecasting that integrates differentially private adjacency sharing with global ASTGNN aggregation to learn secure topology-aware representations.
- **FedSTN** [16]: A federated spatial-temporal network combining recurrent long-term modules, attention-based information sharing, and semantic feature extractors for encrypted vertical FL scenarios.
- **CNFGNN** [35]: A cross-client generative federated GNN that learns a shared generative model while providing client-specific personalization and applying differential privacy to resist inference attacks.
- **UFCL** [36]: A unified federated continual-learning model for streaming spatial-temporal data, featuring a global replay buffer, synthetic trajectory generation, spatio-temporal mixup, and hybrid autoencoder–Siamese objectives.
- **FedCroST** [18]: A personalized federated framework for cross-silo mobility forecasting that learns local spatial-temporal prompts, performs diffusion-based denoising, and aggregates a shared backbone while retaining personalized local modules.

H Client Dropout Robustness

To stimulate unreliable participation, we randomly sample a fraction p of clients at each communication round ($p \in \{1.0, 0.6, 0.3\}$) and train FedCAG under the same total number of rounds in *Mixed Cloud 2*. Table 4 summarizes the resulting mixed-provider performance.

We observe that FedCAG is robust to moderate dropout: reducing participation from $p = 1.0$ to $p = 0.6$ results in only a small increase in MAE, while even aggressive dropout ($p = 0.3$) degrades performance smoothly without catastrophic failure. The combination of graph aggregation, global priors, and regulated/personalized local updates keeps the global model stable despite missing contributions from a subset of clients in each round. Together with the previous non-IID experiments, these results demonstrate that FedCAG maintains strong forecasting accuracy and gracefully degrades under realistic federated conditions.

I Personalization and Head Decomposition Study

To quantify the benefit of decomposing the prediction head into a shared global component and a private adapter, we compare three variants of FedCAG: (i) *Global-only head*, where the adapter is

Table 4: FedCAG Robustness under Random Client Dropout (per-client mean MAE).

Participation rate p	**Avg. MAE**	**Worst MAE**
1.0 (no dropout)	$\mathbf{1.4228}$	$\mathbf{1.6065}$
0.6	1.4654	1.6603
0.3	1.5211	1.7542

disabled, and each client uses only the aggregated global head $\Theta^{\mathrm{head},r}$; (ii) *Local-only head*, where the head is fully local and not aggregated across rounds (no global $\Theta^{\mathrm{head},r}$); and (iii) *Full FedCAG*, which uses the proposed decomposition into a global head $\Theta^{\mathrm{head},r}$ plus a private adapter Ψ_i^r on each client.

For each variant, we train models under the same settings and evaluate per-client forecasting performance on the mixed-provider test split. Figure 3 reports the distribution of per-client MAE as a boxplot across all clients. First, we observe that the global-only head yields the weakest overall accuracy and the largest spread across clients, indicating that a single aggregated predictor cannot fully accommodate heterogeneous workloads. Second, making the head fully local reduces average error but increases variance for clients with limited data or highly noisy traces, as the absence of a global prior leads to overfitting. Third, the proposed FedCAG variant with an aggregated global head and private adapters for clients achieves the lowest MAE and RMSE and substantially reduces per-client variance. In particular, clients with atypical workloads (e.g., GPU-heavy or network-saturated nodes) benefit most from the adapter Ψ_i^r, which can capture local deviations from the global prediction while still leveraging the global head $\Theta^{\mathrm{head},r}$ for generalization.

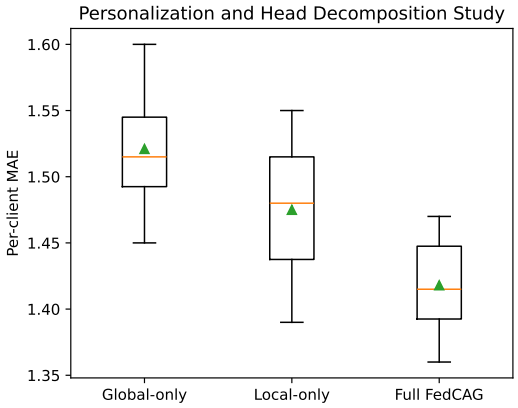


Figure 3: Boxplot distribution of MAE for different heads on Mixed Cloud 2.

J Inference Time

Figure 4 compares the inference latency of centralized and federated workload forecasting methods under different datasets. Among centralized models, Transformer-based approaches such as Autoformer and GPT4TS exhibit the highest latency and the steepest growth with increasing dimensionality, reflecting the quadratic or near-quadratic complexity of attention and auto-correlation mechanisms. Graph-based centralized models (STSGCN, HGCN, and STPGNN) exhibit lower inference costs and more moderate scaling behavior, thanks to localized message passing and sparse structural priors. Nevertheless, inference time still increases noticeably as the number of metrics grows, indicating that centralized deployment remains sensitive to dimensionality even when graph inductive biases are employed.

Federated methods show a broader range of inference efficiency due to differences in model complexity and personalization mechanisms. Recurrent-based baselines (FedGRU and FL-LSTM) maintain low latency but sacrifice expressive power for complex multi-metric interactions. More advanced federated graph models, such as FASTGNN, FedSTN, CNFGNN, UFCL, and FedCroST, incur higher

inference costs due to additional graph operations, generative modules, or diffusion-based components. In contrast, the proposed FedCAG achieves consistently lower inference latency among federated approaches and scales smoothly with dimensionality, benefiting from lightweight personalized heads that enable efficient inference without introducing heavy architectural overhead.

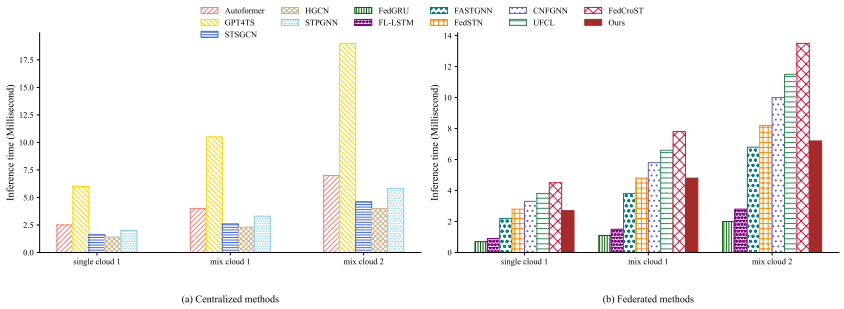


Figure 4: Inference time comparison.

K Additional Experimental Results

Scaling to 50 VMs. Table 5 reports the five methods evaluated in an additional 50-VM experiment. Centralized methods train on pooled VM data, whereas federated methods keep traces local. FedCAG has the lowest error among the evaluated federated methods at this scale; STPGNN remains more accurate with pooled training data. The experiment extends application-level evidence beyond 15 VMs, but does not establish behavior for thousands of clients.

Table 5: Forecasting performance with 50 VMs.

Method	Training	MAE ↓	RMSE ↓
GPT4TS	Centralized	2.3491	4.7982
STPGNN	Centralized	2.3325	4.7783
UFCL	Federated	2.7319	5.5932
FedCroST	Federated	2.6811	5.2423
FedCAG	Federated	**2.4013**	**4.8304**

Run-to-run variation. Table 6 shows the mean and standard deviation for selected methods on *Mixed Cloud 2* over 10 runs. These are standard deviations of run-level MAE and RMSE, not confidence intervals or significance tests. The remaining configurations retain means only in Table 1; their run-level values are needed before complete error bars can be reported.

Table 6: Mean $\pm$ standard deviation over 10 runs on Mixed Cloud 2.

Method	Training	MAE ↓	RMSE ↓
GPT4TS	Centralized	1.2241 ± 0.0109	2.4024 ± 0.0253
STPGNN	Centralized	1.2142 ± 0.0077	2.3592 ± 0.0166
UFCL	Federated	1.7737 ± 0.0104	2.7153 ± 0.0231
FedCroST	Federated	1.6374 ± 0.0098	2.6179 ± 0.0192
FedCAG	Federated	$\mathbf{1.4228 \pm 0.0111}$	$\mathbf{2.5841 \pm 0.0242}$

L Mechanism-Level Ablation

Table 7 complements the graph-view and objective ablations in the main paper. Removing the private adapter has the largest observed effect; removing federated graph priors has the next largest effect. Thus, the contribution lies in the coupled graph-sharing and personalization cycle, rather than any claim that its constituent neural operations are individually novel.

Table 7: Mechanism ablation on Mixed Cloud 2. Relative increases are versus full FedCAG.

Variant	MAE	RMSE	ΔMAE	ΔRMSE
Full FedCAG	1.4228	2.5841	–	–
w/o private adapter	1.5173	2.7742	6.64%	7.36%
w/o federated graph priors	1.4923	2.7148	4.89%	5.06%
w/o graph personalization	1.4671	2.6648	3.11%	3.12%
w/o directed-dependency fusion	1.4517	2.6426	2.03%	2.26%
w/o reliability weighting	1.4479	2.6311	1.76%	1.82%
w/o EMA smoothing	1.4368	2.6146	0.98%	1.18%

M Training-Time Breakdown

Tables 8 and 9 report average wall-clock seconds per communication round in the measured settings. Local optimization dominates the measured FedCAG runtime. These timings include validation-only Optuna search; they do not measure network transfer latency or transmitted bytes, so they do not by themselves quantify communication overhead.

Table 8: FedCAG mean runtime per communication round, in seconds.

Component	Single Cloud	Mixed Cloud 1	Mixed Cloud 2
Local client optimization	10.08	18.59	27.96
Local graph construction/refinement	0.70	1.11	1.58
Global-local graph personalization	0.18	0.28	0.42
Model-parameter aggregation	0.41	0.58	0.80
Reliability-weighted graph aggregation	0.12	0.20	0.27
EMA graph smoothing	0.04	0.06	0.08
Optuna validation-only search	1.17	1.98	2.89
Total	12.70	22.80	34.00

Table 9: Mean seconds per round for selected methods under the reported protocol. Centralized rows are training-time references, not communication rounds.

Method	Protocol	Single Cloud	Mixed Cloud 1	Mixed Cloud 2
GPT4TS	Centralized	19.9	36.1	65.5
STSGCN	Centralized	7.2	11.3	20.7
STPGNN	Centralized	9.3	15.7	27.5
FASTGNN	Federated	10.3	18.0	32.2
FedSTN	Federated	14.0	24.1	41.4
UFCL	Federated	21.4	37.6	65.1
FedCroST	Federated	28.3	49.2	85.2
FedCAG	Federated	12.7	22.8	34.0

N Convergence Analysis

We provide a formal convergence discussion for FedCAG. The analysis follows standard non-convex FL arguments, but accounts for the private adapter through a virtual aggregation step. Let the augmented parameter be

$$\hat{\Theta}^r \triangleq \left(\Theta^{\mathrm{bb},r}, \Theta^{\mathrm{head},r}, \Psi^r\right), \tag{24}$$

where the virtual adapter is defined as

$$\Psi^r = \sum_{i=1}^{M} \pi_i \Psi_i^r. \tag{25}$$

The virtual adapter is only used for analysis. In FedCAG, each Ψ_i^r remains private and is never uploaded. For client i, let $F_i(\hat{\Theta})$ denote the local objective, including the prediction loss, contrastive

loss, and regularization term. The virtual global objective is

$$F(\hat{\Theta}) = \sum_{i=1}^{M} \pi_i F_i(\hat{\Theta}), \quad \sum_{i=1}^{M} \pi_i = 1. \tag{26}$$

Assumption 1 (Smoothness and lower boundedness)**.** *For every client i, F_i is L-smooth, i.e.,*

$$F_i(y) \leq F_i(x) + \langle \nabla F_i(x), y - x \rangle + \frac{L}{2}\|y - x\|^2. \tag{27}$$

The global objective is lower bounded by $F^\star$, i.e., $F(\hat{\Theta}) \geq F^\star$.

Assumption 2 (Unbiased stochastic gradients)**.** *For a stochastic gradient $g_i(\hat{\Theta}; \xi)$ sampled on client i,*

$$\mathbb{E}_\xi \left[g_i(\hat{\Theta}; \xi) \right] = \nabla F_i(\hat{\Theta}), \tag{28}$$

and the variance is bounded:

$$\mathbb{E}_\xi \left\| g_i(\hat{\Theta}; \xi) - \nabla F_i(\hat{\Theta}) \right\|^2 \leq \sigma^2. \tag{29}$$

Assumption 3 (Bounded client-global gradient divergence)**.** *For any $\hat{\Theta}$, the deviation between client gradients and the global gradient is bounded:*

$$\sum_{i=1}^{M} \pi_i \left\| \nabla F_i(\hat{\Theta}) - \nabla F(\hat{\Theta}) \right\|^2 \leq \zeta^2. \tag{30}$$

Let $\hat{\Theta}_i^{r,e}$ be the local parameter of client i at local epoch e in round r, with $\hat{\Theta}_i^{r,0} = \hat{\Theta}^r$. The virtual global update can be written as

$$\hat{\Theta}^{r+1} = \hat{\Theta}^r - \eta \sum_{i=1}^{M} \pi_i \sum_{e=0}^{E-1} g_i(\hat{\Theta}_i^{r,e}; \xi_i^{r,e}). \tag{31}$$

Lemma 1 (Local drift bound)**.** *Under Assumptions 1–3, if $\eta \leq 1/(4LE)$, then the local-update bias*

$$b^r = \frac{1}{E} \sum_{i=1}^{M} \pi_i \sum_{e=0}^{E-1} \left[\nabla F_i(\hat{\Theta}_i^{r,e}) - \nabla F_i(\hat{\Theta}^r) \right] \tag{32}$$

satisfies

$$\mathbb{E}\|b^r\|^2 \leq 8L^2\eta^2E^2 \left(\mathbb{E}\|\nabla F(\hat{\Theta}^r)\|^2 + \zeta^2 + \sigma^2 \right). \tag{33}$$

Proof. By L-smoothness in Assumption 1,

$$\left\| \nabla F_i(\hat{\Theta}_i^{r,e}) - \nabla F_i(\hat{\Theta}^r) \right\| \leq L \left\| \hat{\Theta}_i^{r,e} - \hat{\Theta}^r \right\|. \tag{34}$$

The local iterate $\hat{\Theta}_i^{r,e}$ is obtained after e stochastic gradient steps from $\hat{\Theta}^r$. Using Assumptions 2 and 3, the standard local-SGD drift recursion gives

$$\mathbb{E} \left\| \hat{\Theta}_i^{r,e} - \hat{\Theta}^r \right\|^2 \leq 4\eta^2e^2 \left(\mathbb{E}\|\nabla F(\hat{\Theta}^r)\|^2 + \zeta^2 + \sigma^2 \right). \tag{35}$$

Combining (34) and (35), and then averaging over clients and local epochs, yields (33). □

Theorem 1 (Non-convex convergence of virtual FedCAG)**.** *Suppose Assumptions 1–3 hold. Let $\sigma_\pi^2 = \sigma^2 \sum_{i=1}^{M} \pi_i^2$. If $\eta \leq 1/(8LE)$, then the virtual global sequence generated by* (31) *satisfies*

$$\frac{1}{R} \sum_{r=0}^{R-1} \mathbb{E} \left\| \nabla F(\hat{\Theta}^r) \right\|^2 \leq \frac{4 \left(F(\hat{\Theta}^0) - F^\star \right)}{\eta ER} + 8L\eta\sigma_\pi^2 + 32L^2\eta^2E^2(\sigma^2 + \zeta^2). \tag{36}$$

Proof. Define the stochastic noise term

$$\nu^r = \frac{1}{E}\sum_{i=1}^{M}\pi_i\sum_{e=0}^{E-1}\left[g_i(\hat{\Theta}_i^{r,e};\xi_i^{r,e}) - \nabla F_i(\hat{\Theta}_i^{r,e})\right]. \tag{37}$$

By Assumption 2,

$$\mathbb{E}[\nu^r \mid \hat{\Theta}^r] = 0, \tag{38}$$

and

$$\mathbb{E}\|\nu^r\|^2 \le \frac{\sigma_\pi^2}{E}. \tag{39}$$

Using (32) and (37), the update in (31) becomes

$$\hat{\Theta}^{r+1} = \hat{\Theta}^r - \eta E\left(\nabla F(\hat{\Theta}^r) + b^r + \nu^r\right). \tag{40}$$

By the smoothness of F,

$$\begin{aligned}\mathbb{E}F(\hat{\Theta}^{r+1}) \le \mathbb{E}F(\hat{\Theta}^r) &- \eta E\mathbb{E}\left\langle \nabla F(\hat{\Theta}^r), \nabla F(\hat{\Theta}^r) + b^r + \nu^r\right\rangle \\ &+ \frac{L\eta^2E^2}{2}\mathbb{E}\left\|\nabla F(\hat{\Theta}^r) + b^r + \nu^r\right\|^2.\end{aligned} \tag{41}$$

Since $\mathbb{E}[\nu^r \mid \hat{\Theta}^r] = 0$, the noise has zero inner product with the global gradient in expectation. Using Young's inequality and Lemma 1, and absorbing the bias term under $\eta \le 1/(8LE)$, we obtain

$$\mathbb{E}F(\hat{\Theta}^{r+1}) \le \mathbb{E}F(\hat{\Theta}^r) - \frac{\eta E}{4}\mathbb{E}\left\|\nabla F(\hat{\Theta}^r)\right\|^2 + 2L\eta^2E\sigma_\pi^2 + 8L^2\eta^3E^3(\sigma^2+\zeta^2). \tag{42}$$

Summing (42) from $r=0$ to $R-1$ gives

$$\frac{\eta E}{4}\sum_{r=0}^{R-1}\mathbb{E}\left\|\nabla F(\hat{\Theta}^r)\right\|^2 \le F(\hat{\Theta}^0) - F^\star + 2L\eta^2ER\sigma_\pi^2 + 8L^2\eta^3E^3R(\sigma^2+\zeta^2). \tag{43}$$

Dividing both sides of (43) by $\eta ER/4$ proves (36). □

Discussion. Theorem 1 shows that virtual FedCAG converges to a stationary point of the augmented objective in the standard non-convex sense. The first term in (36) decreases with the number of communication rounds. The second term comes from stochastic gradient noise. The third term captures the effect of local updates and client heterogeneity through ζ^2. When the learning rate decays with R, the average gradient norm approaches zero under bounded variance and bounded client-global gradient divergence. Because the actual global parameters $(\Theta^{\mathrm{bb},R}, \Theta^{\mathrm{head},R})$ are contained in the virtual parameter $\hat{\Theta}^R$, the convergence of the virtual sequence provides the convergence justification for the shared FedCAG model, while the private adapters remain local personalization components.

O Limitations and Discussion

FedCAG studies privacy-preserving workload forecasting in heterogeneous multi-cloud environments, but several limitations remain. Although FedCAG does not share raw telemetry, uploaded model parameters and graph priors may still leak partial information about local workload patterns, so practical deployments should consider secure aggregation, differential privacy, graph perturbation, or trusted execution environments. In addition, the proposed causality graph should be interpreted as a causality-inspired directional dependency prior rather than evidence of identifiable causal effects, since hidden confounders and distribution shifts may affect graph estimation. Our experiments use real traces from Tencent Cloud and Alibaba Cloud through the MUCEP platform, but larger deployments involving more providers, application-level traces, and proprietary scheduling policies may exhibit additional heterogeneity. Finally, FedCAG introduces graph construction and graph aggregation overhead beyond standard parameter averaging, and future work can explore compressed graph updates, asynchronous aggregation, and stronger privacy-preserving structural sharing.

Table 10: Key Notations in FedCAG.

Symbol	Description
M	Number of clients (cloud sites/tenants/VM groups) indexed by i.
t	Timestamp index.
n	Number of system metrics.
$X_i^t \in \mathbb{R}^n$	Metric vector of client i at time t.
w	Sliding window length.
$X_i^{t-w+1:t}$	Input window $\{X_i^{t-w+1}, \ldots, X_i^t\}$ for forecasting.
X_i^{t+1}	Next-step prediction target.
R	Number of federated communication rounds indexed by r.
E	Number of local training epochs per round indexed by e.
$\mathbf{A}_i^r \in \mathbb{R}^{w\times w}$	Temporal graph (intra-window temporal adjacency) for client i in round r.
$\mathbf{B}_i^r \in \mathbb{R}^{n\times n}$	Spatial graph (undirected metric interactions) for client i in round r.
$\mathbf{C}_i^r \in \mathbb{R}^{n\times n}$	Causality graph (directed lead-lag influence) for client i in round r.
$\hat{\mathbf{A}}_i^r, \hat{\mathbf{B}}_i^r, \hat{\mathbf{C}}_i^r$	Personalized graphs after blending global and local priors for client i in round r.
$\mathbf{A}^r, \mathbf{B}^r, \mathbf{C}^r$	Global graph priors aggregated by the server in round r.
$\tilde{\mathbf{A}}^r, \tilde{\mathbf{B}}^r, \tilde{\mathbf{C}}^r$	Reliability-weighted aggregated graphs before EMA smoothing in round r.
Θ_i^r	Parameters for client i in round r, include backbone $\Theta_i^{\mathrm{bb},r}$ and prediction head $\Theta_i^{\mathrm{head},r}$.
Θ^r	Global model parameters aggregated by the server in round r.
Ψ_i^r	Private adapter for client i in round r (local, never uploaded).
π_i	Weight for model aggregation for client i, $\sum_{i=1}^M \pi_i = 1$.
ρ_i^r	Reliability weight for graph aggregation for client i in round r, $\sum_{i=1}^M \rho_i^r = 1$.
α_i^r	Personalization coefficient controlling global-local graph blending for client i in round r.
β	EMA smoothing weight for global graph priors.
$h_i^{\mathrm{intra},r}$	Intra-temporal embedding encoded from $\hat{\mathbf{A}}_i^r$ for client i in round r.
$h_i^{\mathrm{inter},r}$	Inter-metric embedding encoded from $(\hat{\mathbf{B}}_i^r, \hat{\mathbf{C}}_i^r)$ for client i in round r.
c_i^r	Pooled causality descriptor from $\hat{\mathbf{C}}_i^r$ via $\phi(\cdot)$ for client i in round r.
$h_i^{\mathrm{fuse},r}$	Fused embedding via MHA over $[h_i^{\mathrm{intra},r} \| h_i^{\mathrm{inter},r} \| c_i^r]$ for client i in round r.
$\mathcal{L}_i$	Local objective (MSE + InfoNCE + regulation) for client i.
λ_1, λ_2	Weights for InfoNCE and regularization terms.

P Broader Impact

This work studies federated workload forecasting for multi-cloud systems and may support more reliable cloud services by improving autoscaling, resource allocation, and service-level management without centralizing fine-grained telemetry. More accurate forecasting can reduce service disruptions and resource waste, which may benefit both cloud operators and end users. However, forecasting errors may still cause under-provisioning, over-provisioning, or unstable operational decisions if predictions are used without monitoring or human oversight. Federated learning also does not eliminate all privacy risks, since model updates and graph structures may reveal information about local workloads. FedCAG is intended as a research framework for privacy-preserving system-level forecasting; deployment should include privacy protection, access control, failure handling, and validation under realistic cloud operating conditions.

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: The abstract and introduction accurately reflect the paper's contributions and scope.

   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: We discuss limitations in Appendix O.

   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: We provide a theoretical analysis in Method 3.5 and Appendix N.

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: We provide implementation in the Appendix F.

   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: Code is available at supplemental material.

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: Yes, we provide details in Appendix F and E.

   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: [No]

   Justification: For each experimental setting, we report the average performance. However, we do not currently report error bars, confidence intervals, or statistical significance tests.

   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: We provide details in Appendix F.

   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 reviewed the NeurIPS Code of Ethics and confirm that the research conducted in this paper conforms to it.

   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: Broader impacts are discussed in Appendix P

    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: We use our private datasets.

    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: We have properly cited and credited the existing assets used in this work.

    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:[N/A]

    Justification: We use private datasets.

    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 crowdsourcing nor research with human subjects.

    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: This work 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.