entropy

MDPI

Article

FedTIP: Communication-Efficient Federated Temporal Prompting for Few-Shot Dynamic Graph Adaptation

Xijun Wu and Xinming Zhang *

School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China; wuxijun@mail.ustc.edu.cn

* Correspondence: xinming@ustc.edu.cn

Abstract

Temporal interaction streams are often distributed across data owners, while downstream labels and communication budgets remain limited. Existing temporal graph prompts typically adapt at a centralized trainer, whereas federated graph methods commonly exchange larger trainable states. This paper presents FedTIP, a prompt-level federated method for client-local few-shot adaptation of temporal interaction graphs. A centralized historical stage constructs a temporal knowledge bank whose encoder is frozen during downstream federation. Clients optimize interaction-conditioned prompts on local supervised events and transmit a compact prompt-side message. Prototype anchoring and a reliability rule based on support coverage and update magnitude combine heterogeneous client updates, while prompt-side proximal regularization and stage-scoped negative sampling preserve the temporal protocol. Experiments on Wikipedia, Reddit, and MOOC cover temporal node classification and transductive and inductive link prediction. FedTIP records the highest mean AUC-ROC in the nine reported task–dataset cells, with gains of 1.83–14.69 points over the strongest federated baseline in each cell. Its measured cumulative float32 uplink is 0.40 MiB per downstream task, 64.6–99.6% below the evaluated baselines.

Keywords: federated learning; temporal interaction graphs; communication efficiency; prompt tuning; few-shot learning; dynamic graph adaptation

1. Introduction

Online services generate continuous streams of timestamped interactions among users, content, courses, and other entities. Temporal graph models preserve the order and timing of these events: JODIE and DyRep model evolving node states [1,2], whereas TGAT and TGN use temporal attention or memory-based message passing [3,4]. Their evaluation must also respect what was observable at each time. Surveys and benchmark studies identify chronological splitting, inductive status, and time-valid negative sampling as central concerns [5–7]. Thus, adapting a temporal graph model is constrained by both scarce downstream supervision and the history available before each prediction.

In many deployments, organizations retain their own downstream evidence. Federated averaging learns from decentralized data through iterative aggregation of local model updates [8], and federated graph learning extends this access pattern to graph-structured data [9,10]. Communication can be reduced by jointly optimizing model compression and resource allocation [11] or by transmitting sparse gradients from heterogeneous devices [12]. Knowledge distillation offers another compact exchange mechanism for statistically heterogeneous clients [13]. These studies motivate the setting considered here: the client-owned

Check for updates

Academic Editors: Lihua Wang and Rongxing Lu

Received: 28 July 2026
Revised: 24 August 2026
Accepted: 1 September 2026
Published: 4 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

observations form a chronologically evolving interaction stream, only a few labeled events are available for each downstream task, and repeated uplink transmission is constrained.

Pre-training and prompting provide a natural starting point. Prompt tuning was introduced as a parameter-efficient way to adapt a frozen model [14]. Graph and dynamic graph pre-training learn reusable representations from historical structure [15–17]; GraphPrompt, GPF, and All-in-One then adapt graph encoders through compact prompts [18–20]. Temporal prompt methods introduce time-aware or node–time conditional adaptation [21,22], but their downstream support is normally available to a centralized trainer.

Federated graph learning takes the complementary view that downstream observations remain distributed. Static or snapshot-based approaches exchange predictions or graph model parameters [23–25]. Federated dynamic-graph systems exchange temporal–spatial states, dynamic graph parameters, or task-specific temporal representations [26–28]; FedLink formulates federated temporal link prediction in a non-archival NeurIPS workshop manuscript [29]. These two lines of research leave a specific adaptation problem unresolved. Temporal prompting generally assumes centralized access to downstream support, whereas federated dynamic-graph methods coordinate substantially larger model or representation state. Simply applying FedAvg to a compact prompt does not resolve the resulting evidence imbalance: a fixed prompt cannot respond to event-specific partners, time intervals, and pre-event states, and a client with only a few chronological support events may observe one class or produce an update with an atypical magnitude. The challenge is therefore not only to reduce trainable parameters, but also to determine how event-conditioned updates and sparse class evidence should influence shared adaptation.

Figure 1 illustrates the representation side of this problem. Continuous-time events differ in their pre-event node states, interaction partners, and time intervals, whereas a fixed task prompt applies one modulation direction to every event. An interaction-conditioned adapter can instead derive feature and temporal gates from each observable event context. On the aggregation side, client updates must also be interpreted in light of the sparse and potentially single-class support from which they were estimated.

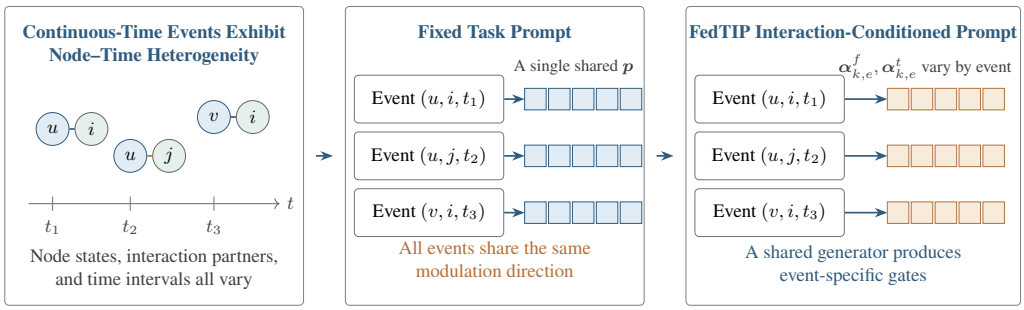


Figure 1. Fixed and interaction-conditioned prompts for client-local temporal adaptation. FedTIP produces event-specific feature and temporal gates from the pre-event interaction context.

FedTIP addresses these coupled representation and aggregation problems by federating an interaction-conditioned adapter over a frozen temporal knowledge bank. Its key mechanism is a coverage-aware reliability rule that combines local label coverage with prompt update magnitude and applies the resulting weights consistently to both prompt deltas and class-anchor updates. Consequently, sparse client evidence influences the shared prompt and prototype geometry through the same reliability semantics. The structured uplink further admits a replace-one sensitivity bound for class prototypes and an additive sufficient-statistics decomposition for secure summation, while raw downstream support events remain with their owners.

FedTIP builds on frozen temporal encoding, proximal regularization, prototypical objectives, and federated averaging; its methodological novelty lies in jointly governing event-conditioned prompt updates and class-anchor geometry through coverage-aware reliability under chronological few-shot support.

The main contributions are summarized as follows:

- We formulate chronologically constrained federated temporal prompt adaptation and instantiate it with a frozen temporal knowledge bank and client-side interaction-conditioned adapters. Raw downstream support events remain with their owners, while recurrent federation exchanges structured prompt-side messages rather than the full temporal model.
- We introduce coverage-aware joint prompt–prototype aggregation. A reliability rule combines local label coverage and prompt update magnitude, and the resulting weights govern both prompt deltas and class-anchor contributions, coupling representation adaptation with prototype alignment under sparse and class-imbalanced client support.
- We characterize the structured message interface through a replace-one prototype-sensitivity bound and an additive sufficient-statistics decomposition, establishing compatibility with sensitivity-calibrated prototype release and secure summation.
- We evaluate FedTIP on three temporal interaction datasets and three task families. It obtains the highest reported mean AUC-ROC in all nine same-access cells among the evaluated methods; matched TGAT/TGN comparisons, component ablations, hyperparameter and partition analyses, and communication measurements isolate the behavior and cost of the proposed mechanism.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents the system model and problem formulation. Section 4 describes FedTIP. Section 5 presents the experimental setup, results, and limitations. Section 6 concludes the paper.

2. Related Work

2.1. Temporal Interaction Modeling and Prompt Adaptation

Dynamic graph representation learning differs in how time is exposed to the encoder. Discrete-time methods transform an event stream into snapshots and apply static or recurrent graph encoders. GCN and GAT provide representative spatial operators [30,31], while ROLAND evolves node embeddings across snapshots [32]. Continuous-time DGNNs operate closer to the interaction stream. JODIE and DyRep model coupled trajectories or point-process dynamics [1,2]. TGAT and TGN use temporal attention or memory states [3,4], while TREND and CAW capture event-node dynamics or causal walks [33,34]. Surveys establish the need for chronological and inductive evaluation [5]; benchmark studies specify time-valid splits and negative-sampling protocols [6,7].

Pre-training reduces dependence on downstream labels by learning reusable graph representations. DDGCL and CPDG exploit temporal regularities in historical streams [16,17]. Prompt learning changes the unit of adaptation: GraphPrompt, GPF, and All-in-One use compact graph prompts [18–20], and surveys organize these methods as parameter-efficient graph adaptation [35,36]. Temporal prompting introduces time-aware or node–time conditional prompts [21,22].

2.2. Communication-Efficient Federated Adaptation

FedAvg establishes the iterative model-averaging baseline for decentralized learning [8]. Communication-aware variants reduce transmitted states through model compression [11], sparse gradients [12], or distillation [13].

Statistical heterogeneity motivates changes to aggregation. FedProx constrains local drift, SCAFFOLD reduces client update variance, and FedNova corrects objective inconsistency under unequal local computation [37–39]. FedDyn adds dynamic regularization, while FedBN retains local normalization statistics under feature shift [40,41]. FedProto establishes a prototype-level federation pattern in which clients upload class prototypes, the server aggregates them, and global prototypes return to clients [42]. FedHide subsequently characterizes leakage from shared class prototypes and replaces true prototypes with proxy representations [43]. Differentially private prototype learning further shows how a frozen public encoder and private class representations can be coupled through sensitivity-calibrated prototype release [44]. Ditto relates regularized local adaptation to personalized federation [45].

2.3. Federated Graph and Dynamic Graph Learning

Federated graph learning makes graph ownership and partitioning explicit. Surveys organize the field around data privacy, graph heterogeneity, and structured collaboration [9,10]. FedGNN addresses privacy-preserving recommendations [23], FedSage+ generates missing cross-client neighbors [24], and FedGCN studies communication–convergence trade-offs for graph convolution [25]. FedGTA uses topology-aware aggregation [46]; FedGraphNN and FedGraph provide benchmark infrastructure [47,48].

Federated dynamic graph learning provides the nearest temporal baseline family. Feddy and STFL study dynamic GNN federation or temporal–spatial graph learning [26,27]. FedDGL models heterogeneous temporal evolution [28], and FedLink addresses federated dynamic link prediction [29].

Taken together, prior studies address temporal prompting, communication-efficient federation, prototype aggregation, and federated dynamic graph learning largely as separate problems. They do not jointly model event-conditioned prompt adaptation and class-prototype aggregation when chronological supervision is both client-owned and extremely sparse.

3. System Model and Problem Formulation

FedTIP considers a coordinating server and independently owned clients under a four-part access model. Historical interactions are available centrally for pre-training; sampled downstream support events are client-owned and used for local adaptation; a global downstream validation segment is visible to the server for model selection; and a global test segment is visible to the server only for final scoring. The locality scope covers the few-shot downstream support sets within this centralized pre-training and evaluation protocol. The objective is to adapt a temporal interaction model while keeping those support events with their owners, respecting chronological visibility, and measuring cumulative uplink payload.

3.1. Temporal Interaction Graph

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a temporal interaction graph represented as a chronologically ordered event stream

$$\mathcal{E} = \{e_n = (u_n, i_n, t_n, y_n, \mathbf{x}_n)\}_{n=1}^{N}, \quad t_1 \leq t_2 \leq \cdots \leq t_N. \tag{1}$$

Here u_n is a source-side node, i_n is a destination-side node, t_n is the interaction timestamp, y_n is an optional dynamic state label associated with the event or source node, and $\mathbf{x}_n \in \mathbb{R}^{d_e}$ is an edge feature vector. For any time t, $\mathcal{H}^{t^-}$ denotes the events observable strictly before t. Before training, the manifest assigns every raw event to one of six mutually exclusive outcomes: historical pre-training, pre-training validation,

an observed downstream supervised pool $\mathcal{E}_{\text{sup}}$, an inductive-node exclusion from the downstream training candidate window, downstream validation $\mathcal{E}_{\text{val}}$, or downstream test $\mathcal{E}_{\text{test}}$. The inductive test set is a flagged subset of $\mathcal{E}_{\text{test}}$ involving a typed user or item node removed from $\mathcal{E}_{\text{sup}}$. Earlier client updates and training negatives draw exclusively from information available before validation and test. The global downstream validation segment selects method-specific hyperparameters and is substantially larger than the 30-event support set; this stronger server-visible resource is part of the reported protocol.

3.2. Deployment and Communication Model

The server coordinates K clients and maintains the shared model state. Before the first downstream task, it deploys the frozen temporal bank to participating clients. In round r of task τ, it broadcasts the current adapter and anchor prototypes and receives the FedTIP uplink message

$$\mathcal{M}_k^{(r)} = \left(\Delta\Omega_k^{(r)}, \{\mathbf{q}_{k,c}^{(r)}, n_{k,c}\}_{c\in\mathcal{Y}_k}, b_k, \|\Delta\Omega_k^{(r)}\|_2\right). \tag{2}$$

The first component is the prompt-adapter delta; the remaining components are class prototype means and counts, a label-coverage score, and the update norm. This tuple constitutes the FedTIP uplink message. Raw downstream support events remain with their assigned clients, while the centrally constructed frozen temporal encoder is deployed separately before downstream adaptation.

The system abstraction assumes that every selected active client completes its local computation and upload within a round. Communication accounting sums float32 values transmitted from clients to the server over all active clients and rounds for one downstream task. Cumulative uplink payload provides the common cross-method measurement, while Section 5.4 analyzes initialization and recurrent downlink from each method's exchanged objects and communication cadence.

3.3. Federated Few-Shot Adaptation

There are K clients $\mathcal{C} = \{1, \ldots, K\}$. In the main user-iid profile, the split manifest first fixes a user-ownership map $\pi_U : \mathcal{V}_U \to \mathcal{C}$ and then assigns every downstream event $e = (u, i, t, y, \mathbf{x})$ to $\pi(e) = \pi_U(u)$. Consequently, all events of one user have exactly one client owner. Client k owns a supervised pool $\mathcal{E}_{\text{sup},k}$, and the client pools form a disjoint union of $\mathcal{E}_{\text{sup}}$. Label-skewed and activity-skewed profiles change the user-to-client map but retain the same pre-training-time materialization and method-independent ownership rule. For a downstream task τ, an active client set $\mathcal{A}_\tau \subseteq \mathcal{C}$ is selected from the manifest. Each active client receives a small local support set

$$\mathcal{D}_{\tau,k} \subseteq \mathcal{E}_{\text{sup},k}. \tag{3}$$

The support rows and their labels remain inside the assigned client boundary. The manifest also assigns validation and test rows to their user owners; the disjoint client views form the corresponding global segments, whereas model selection and final scoring use their unions at the server. This definition is method-agnostic: a baseline may adapt a prediction head, a prompt, or a larger model state, but every method follows the same support ownership, active-client set, chronological split, and common server-side evaluation segments.

3.4. Evaluation Tasks

The evaluation contains three task families. Temporal node classification predicts a state label for the source-side node at the interaction time. Transductive link prediction evaluates later interactions among nodes observed before the downstream test stage.

Inductive link prediction restricts scoring to test events containing at least one typed node excluded from the downstream supervised pool. Each positive link example is paired with one negative destination. A negative event (u, j, t) is valid only if the interaction is absent at time t under the stage-specific history allowed by the protocol. All methods use the same deterministic stage-valid sampling rule and task/split seed. We materialize this rule for all 9000 task-support events, and Section 5.1 reports the resulting collision checks.

3.5. Trust and Privacy Boundary

Attack-oriented multimedia-security studies evaluate exposed intermediate structures under explicit adversarial capabilities; for example, video cryptanalysis considers chosen-plaintext, known-plaintext, and chosen-ciphertext attacks against internal encryption transformations [49]. Following this principle, FedTIP's threat model specifies message visibility and manipulation surfaces for prompt and prototype disclosures.

The protocol assumes that the coordinator and clients execute the prescribed optimization and reporting rules. The confidentiality analysis treats the coordinator as honest-but-curious after it receives the structured uplink messages, and the integrity analysis treats prompt deltas, prototype summaries, counts, coverage values, and update norms as possible manipulation surfaces. Raw downstream support events remain with their manifest-assigned owners. The coordinator observes $\mathcal{M}_k^{(r)}$; a participating client observes the frozen bank, the current adapter, and server-aggregated anchor prototypes. Client-specific prototypes are used to construct the global anchors and are not redistributed to other clients. The bank broadcast defines the reverse disclosure path because its centrally derived historical statistics and negative-sampling support are visible to participating clients.

The prototype message admits a direct sensitivity analysis. Let the representations used in class c be clipped so that $\|\tilde{\mathbf{h}}_{k,e}\|_2 \le C_{\text{emb}}$, and consider fixed-size replace-one adjacency for $\mathcal{D}_{k,c}$. Since

$$\mathbf{q}_{k,c} = \frac{1}{n_{k,c}} \sum_{e \in \mathcal{D}_{k,c}} \tilde{\mathbf{h}}_{k,e}, \tag{4}$$

its ℓ_2-sensitivity is bounded by

$$\Delta_2(\mathbf{q}_{k,c}) \le \frac{2C_{\text{emb}}}{n_{k,c}}. \tag{5}$$

The bound decreases with the local class count. For $n_{k,c} = 1$, the prototype equals the clipped representation of that event, making the small-count case the most informative message. This result formalizes the prototype exposure identified in prototype-sharing studies [43]. Across the 500 task–client pairs per dataset, the realized supports form 604, 602, and 625 local class prototypes on Wikipedia, Reddit, and MOOC. Of these, 104 (17.22%), 102 (16.94%), and 118 (18.88%) are single-event prototypes. These counts characterize how often the sensitivity bound operates in its highest-exposure small-support regime.

FedTIP's weighted prompt and anchor updates can be expressed through four additive sufficient statistics:

$$\begin{aligned} \mathbf{S}_\Omega &= \sum_{k \in \mathcal{A}_\tau} n_k r_k \Delta\Omega_k, & Z_\Omega &= \sum_{k \in \mathcal{A}_\tau} n_k r_k, \\ \mathbf{S}_c &= \sum_{k: c \in \mathcal{Y}_k} n_{k,c} r_k \mathbf{q}_{k,c}, & Z_c &= \sum_{k: c \in \mathcal{Y}_k} n_{k,c} r_k. \end{aligned} \tag{6}$$

The server updates use $\mathbf{S}_\Omega / Z_\Omega$ and $\mathbf{S}_c / Z_c$; therefore, the aggregation rule is compatible with secure summation of the numerators and denominators [50] without changing the client objective or prompt parameterization. In the main profile, these statistics combine

the five active clients selected for each task, and clients receive the resulting global adapter and anchors. Equation (5) also gives a differential-privacy-compatible prototype map

$$\tilde{\mathbf{q}}_{k,c} = \mathbf{q}_{k,c} + \mathcal{N}\left(\mathbf{0}, \sigma^2 \Delta_2^2 \mathbf{I}\right), \tag{7}$$

where $\Delta_2 = 2C_{\text{emb}}/n_{k,c}$ links the perturbation scale to the proved prototype sensitivity. This construction connects the FedTIP message interface to established differential-privacy mechanisms and private prototype learning [44,51].

4. FedTIP Method

FedTIP federates an interaction-conditioned prompt adapter while keeping the temporal encoder frozen. Its central mechanism is coverage-aware joint prompt–prototype aggregation: reliability derived from local label coverage and update magnitude weights both prompt deltas and class-anchor contributions. A temporal knowledge bank supplies historical representations, while the shared reliability semantics govern how heterogeneous few-shot evidence changes the global adapter and prototype geometry.

4.1. Overview and Federated Visibility

Figure 2 organizes FedTIP into three stages. Stage I constructs the frozen bank $\mathcal{B}_0$ from chronological historical interactions. In Stage II, parallel feature and temporal branches map each client-visible event context to an interaction-conditioned representation. In Stage III, clients transmit $\mathcal{M}_k^{(r)}$ from Section 3; the server updates the adapter and anchor prototypes.

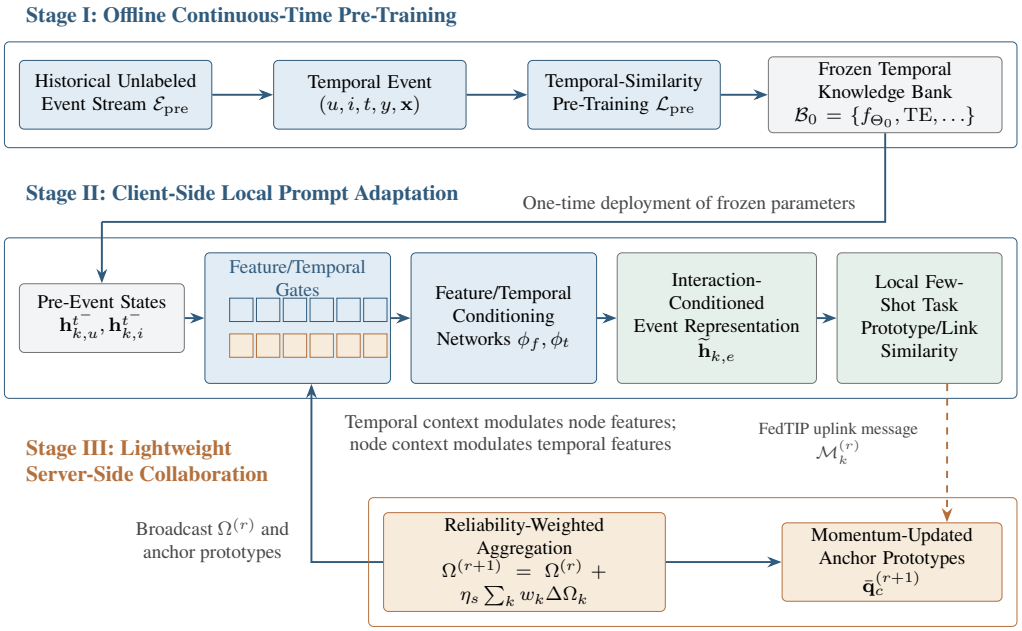


Figure 2. FedTIP workflow. Historical pre-training constructs a frozen temporal bank; client-side adaptation updates event-conditioned prompts; the server aggregates FedTIP uplink messages and updates anchor prototypes.

The encoder, temporal memory, neighbor states, and time encodings are frozen after the centralized historical stage. Downstream visibility and communication follow Section 3; repeated client-to-server transmission is confined to $\mathcal{M}_k^{(r)}$.

Five elements define the method. The bank $\mathcal{B}_0$ contains the frozen temporal encoder and historical metadata. The adapter Ω contains the feature gates, temporal gates, and task template. Local prototypes $\{\mathbf{q}_{k,c}\}$ summarize class geometry, while global anchor prototypes are maintained separately at the server. The reliability score r_k uses label coverage

and update magnitude. The filtered negative index $\mathcal{I}_{\text{neg}}$ is a protocol object, separate from the neural modules.

The bank precedes downstream task sampling, local updates use the frozen bank and preceding events in the assigned task support, and server-side adaptation uses only the FedTIP uplink messages.

4.2. Temporal Knowledge Bank and Interaction Prompting

The temporal knowledge bank defines the shared information available before downstream labels are sampled. It is broader than a parameter checkpoint because it also contains time encodings, historical statistics, and negative-sampling support for permitted stages. The prompt adapter then specializes this shared temporal knowledge to a client-local downstream event without changing the frozen encoder. Anchor prototypes are initialized separately on the server and evolve only during downstream collaboration.

During historical bank construction, the frozen encoder produces temporal node states

$$\mathbf{h}_u^{t^-}, \mathbf{h}_i^{t^-} = f_{\Theta_0}(u, i, t, \mathcal{H}^{t^-}), \tag{8}$$

where $\mathcal{H}^{t^-}$ denotes the chronological history before t in the pre-training stream. The encoder is trained with temporal link prediction on the historical stream. Given a positive interaction (u, i, t), a negative destination is sampled from the pre-training filtered candidate set

$$\mathcal{N}_t^{\text{pre}}(u) = \{j \in \mathcal{V}_I : (u, j, t) \notin \mathcal{H}_{\text{pre}}^t\}, \tag{9}$$

where $\mathcal{V}_I$ is the destination-side node set and $\mathcal{H}_{\text{pre}}^t$ contains pre-training interactions observable up to time t. The filtering rule prevents a same-time or historical positive interaction from being used as a negative example whenever the protocol can detect it. The pre-training objective is

$$\mathcal{L}_{\text{pre}} = -\log \frac{\exp(s(\mathbf{h}_u^{t^-}, \mathbf{h}_i^{t^-})/\tau)}{\exp(s(\mathbf{h}_u^{t^-}, \mathbf{h}_i^{t^-})/\tau) + \exp(s(\mathbf{h}_u^{t^-}, \mathbf{h}_j^{t^-})/\tau)}, \tag{10}$$

where $j \sim \mathcal{N}_t^{\text{pre}}(u)$, $s(\cdot, \cdot)$ is cosine similarity or a bilinear score, and τ is a temperature. The output of this stage is the frozen bank

$$\mathcal{B}_0 = \{f_{\Theta_0}, \text{TE}, \mathcal{S}_{\text{hist}}, \mathcal{I}_{\text{neg}}\}, \tag{11}$$

which contains the encoder, time encoding, historical recency and frequency statistics, and stage-scoped negative-sampling support. The bank provides temporal knowledge to downstream clients without requiring them to learn long-range event dynamics from a handful of labels.

During downstream adaptation, client-side state construction uses the frozen historical bank and the client's task support observed before the current event. For client k in task τ, the event state is written as

$$\mathbf{h}_{k,u}^{t^-}, \mathbf{h}_{k,i}^{t^-} = f_{\Theta_0}(u, i, t, \mathcal{H}_{\text{pre}}^{t^-} \cup \mathcal{H}_{\tau,k}^{t^-}), \tag{12}$$

where $\mathcal{H}_{\text{pre}}^{t^-}$ is the centrally constructed bank context and $\mathcal{H}_{\tau,k}^{t^-} = \{e \in \mathcal{D}_{\tau,k} : t_e < t\}$ contains preceding events in the manifest-assigned task support. Thus, task-specific context is restricted to the client's chronologically preceding support, while the frozen historical bank remains shared.

FedTIP adapts the frozen temporal encoder with a temporal interaction prompt. Generating the prompt from each event context allows the same global adapter to react

differently to recent, old, high-activity, or sparse task support. For each local event $e = (u, i, t, y, \mathbf{x})$, client k constructs

$$\mathbf{z}_{k,e} = [\mathbf{h}_{k,u}^{t^-} \| \mathbf{h}_{k,i}^{t^-} \| \mathrm{TE}(t) \| \mathbf{x} \| \Delta_{k,u}(t) \| \Delta_{k,i}(t)], \tag{13}$$

where $\mathrm{TE}(t)$ is the time encoding, $\Delta_{k,u}(t)$ and $\Delta_{k,i}(t)$ are source- and destination-side recency features available from the frozen bank and preceding task support, and the frozen states are computed under the access rule above. The adapter then generates two gates:

$$\boldsymbol{\alpha}_{k,e}^{f} = \sigma(W_f \phi_f(\mathbf{z}_{k,e}) + \mathbf{b}_f), \tag{14}$$

$$\boldsymbol{\alpha}_{k,e}^{t} = \sigma(W_t \phi_t(\mathbf{z}_{k,e}) + \mathbf{b}_t), \tag{15}$$

where ϕ_f and ϕ_t are bottleneck networks and σ is a sigmoid function. Both branches receive the same event context $\mathbf{z}_{k,e}$ and are evaluated in parallel. Consequently, the temporal encoding and recency terms in $\mathbf{z}_{k,e}$ can modulate the feature gate, while the pre-event node states can modulate the temporal gate, without either bottleneck network consuming the output of the other. The feature gate controls which structural interaction dimensions should be amplified for the current event, while the temporal gate controls how strongly the time encoding enters the downstream representation. The prompted representation is

$$\tilde{\mathbf{h}}_{k,e} = \psi\Big(\boldsymbol{\alpha}_{k,e}^{f} \odot [\mathbf{h}_{k,u}^{t^-} \| \mathbf{h}_{k,i}^{t^-}], \boldsymbol{\alpha}_{k,e}^{t} \odot \mathrm{TE}(t), \mathbf{x}\Big). \tag{16}$$

Here $\psi : \mathbb{R}^{2d_h} \times \mathbb{R}^{d_t} \times \mathbb{R}^{d_e} \to \mathbb{R}^{d_p}$ is a lightweight three-input task template. It concatenates the gated source–destination state, gated time encoding, and edge features and maps them to a d_p-dimensional representation used by the task loss. Only the adapter

$$\Omega = \{W_f, W_t, \phi_f, \phi_t, \mathbf{b}_f, \mathbf{b}_t, \psi\} \tag{17}$$

is trainable and communicated.

4.3. Downstream Objectives and Temporal Regularization

Downstream learning has two layers. The task loss makes the prompt useful for node classification or link prediction. The temporal and federated regularizers keep the client-local update compatible with the shared temporal geometry when only a few events are labeled. For a fixed downstream task τ, we omit the task index and write $\mathcal{D}_k$ for $\mathcal{D}_{\tau,k}$.

For temporal node classification, FedTIP uses the prototypical classification objective [52] and forms a class mean from each client's local support set:

$$\mathbf{q}_{k,c} = \frac{1}{|\mathcal{D}_{k,c}|} \sum_{e \in \mathcal{D}_{k,c}} \tilde{\mathbf{h}}_{k,e}. \tag{18}$$

Here $\mathcal{D}_{k,c}$ denotes the subset of $\mathcal{D}_k$ whose label is c. The same local support events form the prototypes and optimize the support loss below, while validation and test events remain outside this computation. The resulting in-support regularizer can produce optimistic local losses when self-inclusion occurs in a very small class. The local classification loss is

$$\mathcal{L}_k^{\mathrm{nc}} = -\sum_{(e,y) \in \mathcal{D}_k} \log \frac{\exp(s(\tilde{\mathbf{h}}_{k,e}, \mathbf{q}_{k,y})/\tau)}{\sum_c \exp(s(\tilde{\mathbf{h}}_{k,e}, \mathbf{q}_{k,c})/\tau)}. \tag{19}$$

For temporal link prediction, FedTIP uses the same similarity template as pre-training. For any candidate destination v, $\tilde{\mathbf{h}}_{k,u,v,t}$ denotes Equation (16) evaluated with the pre-event states of (u, v) and contains neither the link label nor a post-event state:

$$\mathcal{L}_k^{\mathrm{lp}} = -\sum_{(u,i,j,t)\in\mathcal{D}_k} \log \frac{\exp(s(\tilde{\mathbf{h}}_{k,u,i,t}, \mathbf{h}_{k,i}^{t^-})/\tau)}{\mathcal{Z}_{u,i,j,t}}, \tag{20}$$

$$\mathcal{Z}_{u,i,j,t} = \exp(s(\tilde{\mathbf{h}}_{k,u,i,t}, \mathbf{h}_{k,i}^{t^-})/\tau) + \exp(s(\tilde{\mathbf{h}}_{k,u,j,t}, \mathbf{h}_{k,j}^{t^-})/\tau). \tag{21}$$

The local prototypes can drift sharply when a client receives only a few labeled events. FedTIP therefore maintains server-side anchor prototypes $\bar{\mathbf{q}}_c^{(r)}$ aggregated from previous rounds. Client k regularizes its local prototypes through

$$\mathcal{L}_k^{\mathrm{anchor}} = \sum_{c\in\mathcal{Y}_k} \rho_{k,c}\Big(1 - s(\mathbf{q}_{k,c}, \bar{\mathbf{q}}_c^{(r)})\Big), \tag{22}$$

where $\mathcal{Y}_k$ is the set of labels observed by client k, $n_{k,c} = |\mathcal{D}_{k,c}|$, $n_k = |\mathcal{D}_k|$, and $\rho_{k,c} = n_{k,c}/n_k$. Thus, classes represented by more local support events contribute proportionally more to the anchor penalty.

After each round, the server updates the anchor prototypes from uploaded summaries while raw downstream support events remain with their assigned clients:

$$\hat{\mathbf{q}}_c^{(r+1)} = \frac{\sum_{k:c\in\mathcal{Y}_k} a_{k,c}\mathbf{q}_{k,c}}{\sum_{k:c\in\mathcal{Y}_k} a_{k,c} + \epsilon}, \tag{23}$$

$$\bar{\mathbf{q}}_c^{(r+1)} = \mathrm{norm}\Big((1-\beta)\bar{\mathbf{q}}_c^{(r)} + \beta\hat{\mathbf{q}}_c^{(r+1)}\Big), \tag{24}$$

where $a_{k,c} = r_k n_{k,c}$ is an unnormalized class-specific anchor weight, β controls anchor momentum, and $\mathrm{norm}(\cdot)$ denotes unit normalization.

For downstream link prediction, temporal negatives are sampled from the client-visible candidate set

$$\mathcal{N}_{k,t}^{\mathrm{tr}}(u) = \{j \in \mathcal{V}_I : (u,j,t) \notin \mathcal{H}_{\mathrm{pre}}^t \cup \mathcal{H}_{\tau,k}^t\}, \tag{25}$$

The training filter is constructed from the frozen bank and preceding events in the selected client's task support. Validation and test use the common deterministic stage-valid sampler defined in Section 3. The positive and negative candidates are encoded symmetrically from pre-event states and candidate-specific contexts; labels and future events do not enter either context.

The total client loss is

$$\mathcal{L}_k = \mathcal{L}_k^{\mathrm{task}} + \lambda_p\|\Omega_k - \Omega^{(r)}\|_2^2 + \lambda_a\mathcal{L}_k^{\mathrm{anchor}}, \tag{26}$$

where $\Omega^{(r)}$ is the server prompt at round r and $\mathcal{L}_k^{\mathrm{anchor}}$ aligns local prototypes with server-side temporal anchors. The proximal term follows the same motivation as FedProx in heterogeneous federated optimization, but it is applied only to prompt parameters [37].

4.4. Reliability-Weighted Aggregation and Training Procedure

Aggregation converts local temporal evidence into a shared prompt update. A simple mean gives the same influence to balanced support and to an update formed from one outcome. FedTIP instead computes each round-wise client weight solely from support size, label coverage, and update magnitude.

After local training, each active client sends $\mathcal{M}_k^{(r)}$. Let $p_{k,c} = n_{k,c}/n_k$ for every task label $c \in \mathcal{Y}$, with $p_{k,c} = 0$ when client k has no example of class c. The label-coverage factor is

$$b_k = 1 - \frac{1}{2}\sum_{c\in\mathcal{Y}}\left|p_{k,c} - \frac{1}{|\mathcal{Y}|}\right|. \tag{27}$$

It equals one for a balanced local support distribution and decreases as the support concentrates on fewer outcomes. At round r, the update-norm threshold is determined from the active-client updates as

$$\gamma^{(r)} = \text{median}_{k \in \mathcal{A}_\tau} \left\| \Delta\Omega_k^{(r)} \right\|_2. \tag{28}$$

The reliability factor and normalized aggregation weight are

$$r_k = b_k \min\left(1, \frac{\gamma^{(r)}}{\|\Delta\Omega_k^{(r)}\|_2 + \epsilon}\right), \tag{29}$$

$$w_k = \frac{n_k r_k}{\sum_{j \in \mathcal{A}_\tau} n_j r_j}, \tag{30}$$

Here, $\gamma^{(r)}$ is the median active-client update norm before reliability clipping and $\epsilon = 10^{-8}$ prevents division by zero. The factor n_k accounts for support size; b_k accounts for label coverage; and the clipped norm factor limits unusually large updates. The global prompt is updated by

$$\Omega^{(r+1)} = \Omega^{(r)} + \eta_s \sum_{k \in \mathcal{A}_\tau} w_k \Delta\Omega_k. \tag{31}$$

The server step η_s scales the reliability-weighted prompt update, whereas β remains specific to anchor momentum. A single-class or single-outcome client therefore retains a nonzero weight, but an equally sized balanced client receives greater influence.

Algorithm 1 summarizes the procedure. The chronological split is created once, the temporal encoder is pre-trained centrally, and each downstream task federates only prompt-side adaptation over client-local support events.

Algorithm 1 FedTIP Federated Temporal Interaction Prompting

Require: Chronological events $\mathcal{E}$, client partition, local downstream support sets, rounds R
Ensure: Global prompt adapter Ω and anchor prototypes $\bar{\mathcal{Q}}$

```
Split E into pre-training, supervised-pool, validation, and test segments.
Pre-train temporal encoder f_{Θ_0} on historical link prediction; build B_0 and freeze Θ_0.
Initialize prompt adapter Ω^(0) and anchor prototypes separately on the server.
for each downstream task do
    Select participating clients and construct their local support events from the task manifest.
    Deploy frozen bank parameters and metadata once to participating clients.
    for round r = 0, ..., R − 1 do
        Server broadcasts Ω^(r) and anchor prototypes Q̄^(r).
        for client k in active clients do
            Build event contexts from frozen-bank features and preceding local support events.
            Sample negatives from client-visible N^tr_{k,t}(u) for link prediction.
            Optimize Ω_k with task loss, proximal regularization, and prototype anchoring.
            Upload the FedTIP message M_k^(r).
        end for
        Server computes reliability weights w_k, updates anchors, and obtains Ω^(r+1).
    end for
    Select the configuration on the common server-side validation segment and report the chronological test score.
end for
```

5. Experiments

The experiments evaluate predictive performance, access model effects, and cumulative uplink payload under the system model above. Federated methods share the same split manifest, client visibility, active-client set, and support budget. A matched-backbone comparison separates pooled prompt adaptation, direct FedAvg, and FedTIP; ablation and sensitivity analyses examine the reported components and partition conditions.

5.1. Experimental Setup

5.1.1. Datasets and Temporal Protocol

The data protocol starts from raw temporal event tables and materializes a split manifest before training. Wikipedia and Reddit are user–content interaction graphs with binary dynamic user-state labels; MOOC is a student–course interaction graph with a binary dropout-related state label. Table 1 summarizes graph scale, feature dimensions, label imbalance, and six mutually exclusive chronological partitions. "NN-excl." denotes downstream training-candidate events removed because they touch sampled inductive nodes, while "Ind. test" is a flagged subset of the test stage. Wikipedia and Reddit use 172-dimensional event features; MOOC uses 4-dimensional standardized event features.

Table 1. Dataset statistics and chronological event partitions.

Dataset	Raw Event Statistics							Chronological Manifest Counts						
	Events	Users	Items	Feat. Dim	Classes	Pos. Labels	Pos. Rate	Pre-Train	Pre-Val	Sup. Pool	Down-Val	Test	NN-Excl.	Ind. Test
Wikipedia	157,474	8227	1000	172	2	217	0.001378	110,232	15,747	1810	3150	25,195	1340	11,333
Reddit	672,447	10,000	984	172	2	366	0.000544	470,713	67,244	11,097	13,449	107,592	2352	17,344
MOOC	411,749	7047	97	4	2	4066	0.009875	288,224	41,176	4699	8235	65,879	3536	30,817

The split is chronological and encoded in the manifest, following the temporal visibility convention used by TGAT, TGN, and subsequent temporal graph benchmarks [3,4,7]. The first 70% of each event sequence forms centralized pre-training, and the following 10% forms pre-training validation. Within the final 20%, the first 10% (approximately 2% of the complete stream) is the downstream training-candidate window, the next 10% is global downstream validation, and the remaining 80% is global testing. Sampling typed inductive users and items removes training-candidate events that touch those nodes, leaving supervised pools of 1810 (1.149%), 11,097 (1.650%), and 4699 (1.141%) events for Wikipedia, Reddit, and MOOC. The corresponding excluded counts are 1340, 2352, and 3536. The six manifest outcomes in Table 1 are disjoint and exhaustive for each raw event stream.

The main user-iid manifests use seed 2026 and assign each user to exactly one of ten clients before task generation. This gives 822–823, 1000, and 704–705 users per client on Wikipedia, Reddit, and MOOC. The corresponding client supervised pools contain 102–257, 996–1261, and 357–568 events. Each of the 100 generated tasks then samples exactly 30 supervised temporal interactions from five active clients, following the low-resource temporal prompt protocol of DyGPrompt [22]. The exact 30-event total defines the matched low-supervision comparison and equals 0.0190%, 0.00446%, and 0.00729% of the three raw streams. Uniform allocation together with the forced task-level positive yields 5–7 events per active client. The realized tasks contain 1.04, 1.02, and 1.32 positive state labels on average, respectively, and all 100 tasks per dataset contain both labels. On average, 1.04, 1.02, and 1.25 clients per task contain both labels. A client forms prototypes for its observed label set, and the global server-side scorer computes AUC-ROC. The 100 tasks are generated by repeated sampling from the shared supervised pool.

The sensitivity profile examines client coverage and partition skew on Wikipedia as a representative temporal interaction dataset. It varies the number of active supervised

clients from two to ten and sets the total supervised temporal-event support to $6A$ for A active clients. The user-iid profile allocates this support approximately evenly across active clients. The label-skew profile keeps the total support fixed, but biases state-label composition across clients, so some clients observe mostly one label type. The activity-skew profile also keeps the same total support but makes client activity uneven, so some active clients hold more supervised temporal events than others. All client ownership files, task samples, validation segments, and test segments are materialized before model training and held fixed throughout evaluation.

5.1.2. Baselines

Section 2 gives the baseline taxonomy. In the experiments, FedSage+, FedGCN, and FedGNN receive pre-downstream history-aggregated graph views; Feddy, STFL, FedDGL, and FedLink use their temporal or dynamic adaptation mechanisms. Each implementation is constrained to the same client-local downstream support and chronological visibility. FedLink provides the closest temporal federated comparator for link prediction.

The matched access model comparison uses TGAT [3] and TGN [4] with DyGPrompt [22]. For each backbone, the centralized row pools support events, the direct FedAvg row averages the same client-trained prompt parameters, and the FedTIP row uses the proposed adapter and aggregation under the same client visibility [8].

All federated comparisons share the chronological split, support budget, active-client set, and support ownership. A pre-trained baseline uses the same historical segment as FedTIP; other baselines start from the graph state or temporal features available before downstream supervision. Link prediction uses the historically common binary protocol of one negative per positive [7]. Negative destinations are generated by the same deterministic stage-valid rule and task/split seed for every method. The candidate pool contains destinations observed in pre-training history. The `same_time` rule excludes the positive destination and every true destination paired with the source at that timestamp; a stricter diagnostic mode additionally excludes every historical source–destination pair. Materializing both modes over 3000 support events per dataset gives zero positive collisions, zero same-time true-edge collisions, and zero sampling failures; the stricter mode also gives zero historical-pair collisions. Validation and test use the global chronological segments at the common server-side scorer.

5.1.3. Evaluation Metrics

The primary metric is AUC-ROC for the three task families. Each table entry is the mean and standard deviation over 100 generated task identifiers, with link prediction reported separately on transductive and inductive subsets. Communication is the cumulative float32 uplink per downstream task under the same active-client profile.

5.1.4. Implementation Details

Unless otherwise stated, FedTIP uses a TGN-style temporal memory encoder as the frozen bank; TGAT provides the matched alternative backbone. During downstream adaptation, each active client optimizes only the prompt-side parameters with Adam, a learning rate of 10^{-3}, dropout of 0.1, and weight decay of 10^{-5}; the temporal encoder remains frozen. Federation runs for $R = 20$ rounds with $E = 5$ local epochs per round. Each local epoch uses the complete client support as one batch, corresponding to 5–7 events under the main task manifests. The server step is $\eta_s = 1.0$. Every similarity-softmax loss parameterized by τ in Section 4 uses $\tau = 0.1$.

The reliability rule sets $\gamma^{(r)}$ to the median active-client update norm in round r and uses $\epsilon = 10^{-8}$ for numerical stability. The main mechanism coefficients are $\lambda_p = 10^{-2}$, $\lambda_a = 10^{-1}$, and $\beta = 0.1$. They are selected by one-at-a-time validation sweeps over

$\lambda_p, \lambda_a \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and $\beta \in \{0, 0.01, 0.05, 0.1, 0.5, 0.9, 1\}$, holding the other two coefficients and all fixed implementation settings at their main values. The experimental protocol fixes seed 2026, ten total clients, five active clients, 30 support events, 100 generated tasks, a 1:1 link-positive/negative ratio, deterministic negative-sampling seeds, and float32 communication accounting. Chronological split manifests, user-level ownership assignments, task-support allocations, client-indexed validation/test attribution views, and materialized support-stage negative selections instantiate these choices for all three datasets.

5.2. Main Results

Table 2 reports mean ± standard deviation over the 100 generated few-shot tasks. Federated graph baselines appear above federated dynamic-graph baselines; FedTIP uses the TGN-style frozen encoder. Bold and underlined values mark the highest and second-highest mean in each column.

Table 2. Mean AUC-ROC (%) for node classification and transductive and inductive link prediction; standard deviations are over 100 generated tasks.

Method	Node Classification AUC			Transductive LP AUC			Inductive LP AUC		
	Wiki.	Reddit	MOOC	Wiki.	Reddit	MOOC	Wiki.	Reddit	MOOC
FedSage+	61.84 ± 8.91	54.72 ± 7.36	56.31 ± 5.84	58.76 ± 1.64	62.10 ± 2.27	52.88 ± 1.31	56.21 ± 1.87	59.43 ± 1.52	52.46 ± 2.36
FedGCN	66.33 ± 6.74	58.96 ± 6.29	61.74 ± 4.97	63.58 ± 2.18	69.72 ± 1.96	55.41 ± 1.42	59.84 ± 2.43	64.87 ± 1.74	55.23 ± 2.05
FedGNN	54.69 ± 9.82	62.83 ± 5.67	50.94 ± 5.21	72.15 ± 0.38	66.08 ± 1.45	54.07 ± 0.93	73.46 ± 0.41	68.35 ± 1.16	54.88 ± 1.37
Feddy	63.48 ± 6.51	56.29 ± 7.24	64.02 ± 4.88	61.76 ± 2.84	68.91 ± 2.16	51.62 ± 1.03	58.74 ± 2.61	62.39 ± 2.35	51.87 ± 1.78
STFL	70.16 ± 5.18	67.58 ± 6.93	58.74 ± 5.64	76.88 ± 1.26	82.41 ± 1.13	63.92 ± 2.04	75.31 ± 1.52	79.02 ± 1.36	62.46 ± 2.21
FedDGL	56.42 ± 8.37	64.37 ± 5.95	62.76 ± 5.16	68.53 ± 0.91	59.84 ± 2.73	56.73 ± 0.68	61.08 ± 1.19	57.96 ± 2.48	56.15 ± 2.67
FedLink	71.08 ± 4.76	68.91 ± 5.34	65.38 ± 4.42	82.47 ± 1.58	83.12 ± 1.87	65.33 ± 3.16	81.36 ± 1.43	79.48 ± 2.11	64.22 ± 3.08
FedTIP	**72.91** ± 3.57	**72.42** ± 3.26	**67.64** ± 4.03	**92.86** ± 0.16	**95.31** ± 0.09	**68.64** ± 0.82	**90.81** ± 0.23	**94.17** ± 0.11	**68.28** ± 0.71

As shown in Table 2, FedTIP has the highest reported mean in all nine cells. Relative to the strongest non-FedTIP mean in each column, the observed differences are +1.83, +3.51, and +2.26 points for node classification; +10.39, +12.19, and +3.31 for transductive link prediction; and +9.45, +14.69, and +4.06 for inductive link prediction. FedLink is the nearest non-FedTIP mean in most cells. The node-classification columns have smaller mean differences and larger across-task standard deviations than the link-prediction columns.

5.3. Access Model Comparison

Table 3 holds the prompt family and backbone fixed while changing downstream access and aggregation. Bold and underlined values mark the highest and second-highest means.

Table 3. Mean AUC-ROC (%) for centralized prompting, direct FedAvg, and FedTIP with matched TGAT or TGN backbones.

Method	Node Classification			Transductive LP			Inductive LP		
	Wiki.	Reddit	MOOC	Wiki.	Reddit	MOOC	Wiki.	Reddit	MOOC
TGAT-DyGPrompt (centralized)	**82.64** ± 6.61	73.06 ± 6.32	**77.41** ± 5.21	70.21 ± 0.27	91.02 ± 0.14	54.36 ± 1.04	52.94 ± 0.32	75.71 ± 0.24	53.08 ± 0.95
TGAT-DyGPrompt (FedAvg)	76.48 ± 8.63	67.91 ± 8.41	71.84 ± 7.34	63.72 ± 1.74	84.61 ± 1.29	48.43 ± 2.64	46.94 ± 1.68	69.37 ± 1.91	47.82 ± 2.49
TGAT-FedTIP	80.73 ± 6.58	72.09 ± 6.72	76.31 ± 5.21	68.42 ± 0.24	89.28 ± 0.13	52.37 ± 1.08	51.09 ± 0.28	73.84 ± 0.21	51.72 ± 0.94
TGN-DyGPrompt (centralized)	74.12 ± 3.62	**74.39** ± 3.25	69.54 ± 4.08	**94.01** ± 0.19	**96.46** ± 0.11	**69.91** ± 0.87	**92.63** ± 0.25	**95.38** ± 0.13	**69.42** ± 0.79
TGN-DyGPrompt (FedAvg)	68.36 ± 5.94	68.14 ± 5.21	62.97 ± 6.42	88.17 ± 1.38	90.42 ± 1.12	64.06 ± 2.31	86.11 ± 1.46	89.43 ± 1.03	63.48 ± 2.18
TGN-FedTIP	72.91 ± 3.57	72.42 ± 3.26	67.64 ± 4.03	92.86 ± 0.16	95.31 ± 0.09	68.64 ± 0.82	90.81 ± 0.23	94.17 ± 0.11	68.28 ± 0.71

Table 3 shows that, averaged over the nine cells, direct FedAvg is 5.92 points below centralized TGAT-DyGPrompt and 6.08 points below centralized TGN-DyGPrompt. The corresponding task-family differences are 5.63, 6.28, and 5.87 points for TGAT and 6.19, 5.91, and 6.14 points for TGN (node classification, transductive link prediction, and inductive link prediction).

Relative to direct FedAvg, the FedTIP row is higher by 4.30 points with TGAT and 4.66 points with TGN on average. The task-family differences are 4.30, 4.44, and 4.17 points for TGAT and 4.50, 4.72, and 4.75 for TGN. The remaining mean difference from the centralized row is 1.62 points for TGAT and 1.42 points for TGN.

Table 3 also reveals a marked backbone difference on inductive link prediction. The TGAT means on Wikipedia (52.94) and MOOC (53.08) are close to chance, whereas TGN reaches 92.63 and 69.42 under centralized adaptation and 90.81 and 68.28 with FedTIP. The inductive subset contains nodes excluded from the supervised pool, making backbone choice especially consequential when query-time representations must be formed for unseen nodes from the same 30-event adaptation support.

5.4. Communication Efficiency

We measure client-to-server float32 payload. Let P_m be the number of float32 values uploaded by method m over all active clients and rounds of one downstream task. The measured payload is

$$\mathrm{MiB}_{\mathrm{up}}(m) = \frac{4 \cdot P_m}{1024^2}, \tag{32}$$

The factor four is the number of bytes per float32 value. Method-specific uploaded updates and summaries are counted under the same rule. Figure 3 reports this measured cumulative uplink, while the subsequent equations characterize initialization and recurrent downlink.

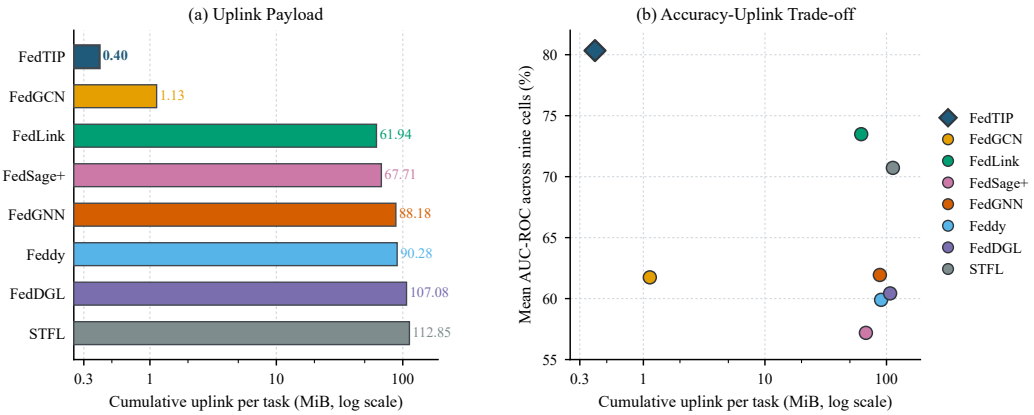


Figure 3. Communication efficiency under measured cumulative float32 uplink. (**a**) Uplink per downstream task, sorted by payload on a logarithmic axis. (**b**) Mean AUC-ROC across the nine task–dataset cells versus the same uplink.

Figure 3a reports 0.40 MiB for FedTIP, 1.13 MiB for FedGCN, and 61.94–112.85 MiB for the other baselines. Within the measured uplink boundary, these values give reductions of 64.6% relative to FedGCN and 99.4–99.6% relative to the remaining methods. FedGCN is the next-lowest uplink method, and its mean AUC-ROC is below FedTIP in each cell of Table 2.

Figure 3b uses the same measured uplink values and places FedTIP at the lowest-uplink and highest-mean position among the evaluated same-access methods. The cell-wise results in Table 2 support this aggregate position: FedTIP has the highest mean in every task–dataset cell.

Downlink has two distinct components. FedTIP deploys the frozen temporal bank once to each newly provisioned client; subsequent rounds broadcast only the global adapter and server-aggregated anchor prototypes. Let N_{init} be the number of clients receiving the bank, $A_{\tau,r}$ the number of active clients in round r of task τ, and R_τ the number of rounds. Its downlink is therefore

$$D^{\downarrow}_{\text{FedTIP}}(T) = N_{\text{init}}|\mathcal{B}_0| + \sum_{\tau=1}^{T}\sum_{r=1}^{R_\tau} A_{\tau,r}\left(|\Omega| + \sum_{c}|\bar{\mathbf{q}}_c|\right). \tag{33}$$

The first term is a reusable initialization cost, whereas the second contains only prompt-side state. FedSage+ [24], FedGCN [25], and FedGNN [23] repeatedly distribute graph model state and, where required, neighborhood or embedding information. Feddy [26] and STFL [27] synchronize dynamic or temporal–spatial model state; FedDGL [28] and FedLink [29] additionally exchange personalized, distillation, or temporal-embedding state. Writing S_m for the recurrent server state of baseline m and $D^{\downarrow}_{m,\text{aux}}$ for its initialization and method-specific auxiliary transfers gives

$$D^{\downarrow}_{m}(T) = \sum_{\tau=1}^{T}\sum_{r=1}^{R_{m,\tau}} A_{\tau,r}|S_m| + D^{\downarrow}_{m,\text{aux}}. \tag{34}$$

For FedTIP, the recurrent downlink is bounded by one adapter and one global-anchor set per active client and remains on the same sub-MiB parameter scale as its 0.40 MiB measured uplink. For baselines that exchange dense model state in both directions, the measured 1.13–112.85 MiB uplinks provide first-order estimates of recurrent model-state downlink under the same client and round schedule, with feature, embedding, and protocol transfers adding method-specific payload. Figure 3 provides the common measured-uplink comparison, and the equations above characterize the corresponding downlink structure. With a cached client cohort, the average FedTIP initialization term decreases as $N_{\text{init}}|\mathcal{B}_0|/T$, while recurrent full-state broadcasts remain proportional to the number of rounds. If $d^{\downarrow}_{m} > d^{\downarrow}_{\text{FedTIP}}$ denotes the corresponding per-task recurrent costs, the symbolic amortization point is $T^{\star}_{m} = \lceil N_{\text{init}}|\mathcal{B}_0|/(d^{\downarrow}_{m} - d^{\downarrow}_{\text{FedTIP}})\rceil$.

5.5. Ablation Study

Table 4 reports absolute AUC-ROC values for the full model, five single-component variants, and a joint prototype–aggregation variant. Each value first averages the 100 generated tasks within a dataset and then assigns equal weight to Wikipedia, Reddit, and MOOC. A cross denotes a removed component, whereas a blank entry denotes that the component is retained.

Table 4. Ablation study on FedTIP design components.

Methods	TKB	ICP	PA	RWA	FTN	Mean AUC-ROC		
						NC	T-LP	I-LP
Variant 1	×					65.88	77.61	76.24
Variant 2		×				67.81	80.72	78.93
Variant 3			×			68.47	81.18	79.84
Variant 4				×		68.13	81.48	80.02
Variant 5					×	70.21	80.06	78.69
Variant 6			×	×		67.49	80.28	78.37
FedTIP						70.99	85.60	84.42

The variants preserve the main protocol and change only the stated mechanism. Removing TKB keeps the encoder architecture and downstream interface but omits histori-

cal self-supervised pre-training before the encoder is frozen. Removing ICP replaces the interaction-conditioned prompt with a static shared prompt. Removing PA sets $\lambda_a = 0$ and suppresses global-anchor exchange and updates while retaining the local class prototypes required by node classification. Removing RWA sets $r_k = 1$, which gives support-weighted FedAvg, $w_k = n_k / \sum_j n_j$, and class-anchor weights $a_{k,c} = n_{k,c}$. Removing FTN uses unfiltered destination sampling in both pre-training and downstream adaptation while still excluding the current positive destination. Variant 6 removes PA and RWA together and keeps TKB, ICP, FTN, and all remaining settings fixed.

As shown in Table 4, removing TKB decreases NC, T-LP, and I-LP by 5.11, 7.99, and 8.18 points relative to full FedTIP, the largest losses in the table. Removing ICP decreases the three task-family means by 3.18, 4.88, and 5.49 points. These patterns connect historical temporal knowledge and event-conditioned prompting to adaptation from sparse supervision. FTN is more task-specific: its removal decreases NC by 0.78 points but T-LP and I-LP by 5.54 and 5.73 points, respectively, consistent with negative destinations entering the link objective directly.

Table 4 further shows that removing PA decreases the three means by 2.52, 4.42, and 4.58 points, while replacing RWA with support-weighted FedAvg decreases them by 2.86, 4.12, and 4.40 points. Their joint removal produces larger losses than either single removal—3.50, 5.32, and 6.05 points—but smaller losses than the sum of the two single-component effects. To quantify this pattern, for task family τ we define

$$I^{\tau}_{\mathrm{PA,RWA}} = \Delta^{\tau}_{\mathrm{PA+RWA}} - \Delta^{\tau}_{\mathrm{PA}} - \Delta^{\tau}_{\mathrm{RWA}}, \tag{35}$$

where $\Delta^{\tau}_{X} = M^{\tau}_{\mathrm{full}} - M^{\tau}_{-X}$. The resulting interaction contrasts are -1.88, -3.22, and -2.93 for NC, T-LP, and I-LP. The negative contrasts describe sub-additive overlap between PA and RWA, consistent with both mechanisms acting through the server-side prototype and client update aggregation path.

5.6. Core Hyperparameter Sensitivity

Figure 4 examines the three coefficients that directly regulate downstream adaptation. Each one-at-a-time sweep uses the same 100 Wikipedia task identifiers under the main low-supervision setting, with the other two coefficients and all protocol settings fixed at their main values. The error bars quantify task-level dispersion across these controlled comparisons.

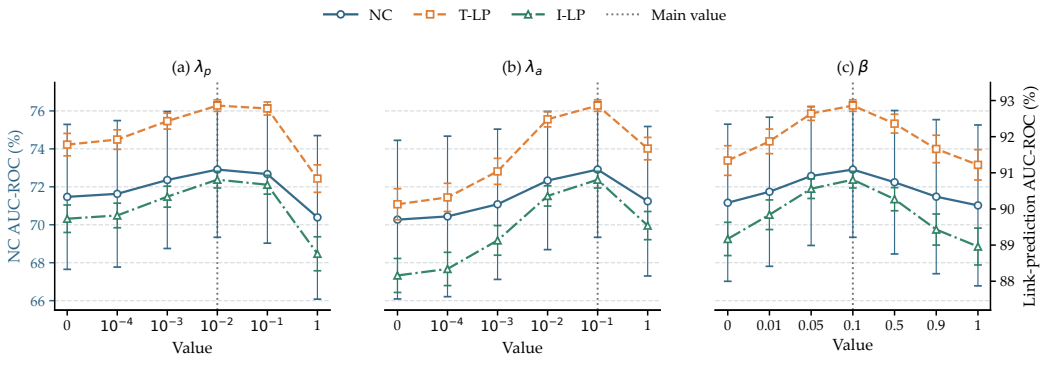


Figure 4. One-at-a-time sensitivity of core FedTIP hyperparameters on Wikipedia under the main low-supervision setting. Curves show mean AUC-ROC (%), error bars show the standard deviation over 100 generated tasks, and vertical dotted lines mark the main configuration. NC uses the left axis; T-LP and I-LP use the right axis.

Figure 4 shows that the selected $\lambda_p = 10^{-2}$ improves NC, T-LP, and I-LP by 1.44, 1.08, and 1.08 points over $\lambda_p = 0$, and by 2.52, 2.02, and 2.05 points over $\lambda_p = 1$. This

interior optimum balances client-specific adaptation against proximal control: removing the term permits greater local drift, whereas an excessive coefficient suppresses useful prompt updates. Similarly, $\lambda_a = 10^{-1}$ improves the three tasks by 2.64, 2.73, and 2.65 points over no prototype anchoring and by 1.67, 1.19, and 1.27 points over $\lambda_a = 1$, indicating that moderate class-level alignment is useful without forcing heterogeneous clients too tightly toward the global anchors.

For anchor momentum, $\beta = 0.1$ exceeds $\beta = 0$ by 1.75, 1.52, and 1.64 points and $\beta = 1$ by 1.89, 1.64, and 1.85 points on NC, T-LP, and I-LP. The result favors strong historical smoothing with a modest current-round update: $\beta = 0$ completely preserves the previous anchor and prevents incorporation of current client evidence, whereas $\beta = 1$ follows the current-round aggregate without historical carry-over. Across all three sweeps and task families, the selected values attain both the highest mean and the smallest observed task-level standard deviation.

5.7. Client Coverage, Support Scaling, and Partition Skew

Figure 5 varies the active-client count A from two to ten on Wikipedia and sets total support to $6A$ events. The horizontal trend therefore measures the joint effect of broader client coverage and increased supervision. At fixed A, user-iid, label-skew, and activity-skew use the same support count, isolating the effect of the partition profile.

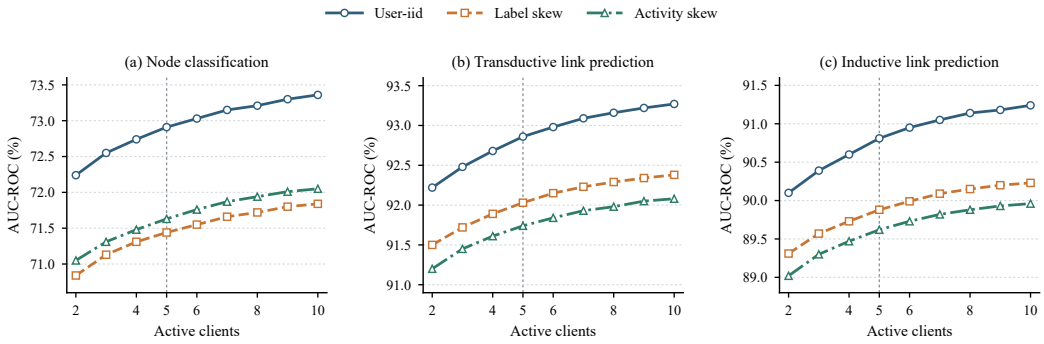


Figure 5. Wikipedia client coverage, support scaling, and partition-skew analysis; all three profiles use $6A$ support events at active-client count A.

As shown in Figure 5, under user-iid partitioning, increasing A from two to ten raises AUC-ROC from 72.24 to 73.36 for node classification, 92.22 to 93.27 for transductive link prediction, and 90.10 to 91.24 for inductive link prediction. The five-client points (72.91, 92.86, and 90.81) match the Wikipedia FedTIP row in Table 2. Beyond five clients, each curve gains less than 0.5 points.

Figure 5 further shows that, at five active clients, label skew is 1.47, 0.83, and 0.93 points below user-iid, while activity skew is 1.28, 1.12, and 1.19 points below. At ten clients, the respective gaps are 1.52, 0.89, and 1.01 points for label skew and 1.31, 1.19, and 1.28 for activity skew.

5.8. Limitations and Scope

The study has three principal scope boundaries. First, historical pre-training and downstream validation are centralized; the locality scope covers downstream support events. Second, the cross-method quantitative bars cover cumulative float32 uplink under complete participation of the selected clients. Downlink is characterized from each method's exchanged objects and communication cadence: FedTIP combines one-time frozen-bank deployment with recurrent adapter-and-anchor broadcasts, whereas the compared full-state methods repeatedly distribute substantially larger model or auxiliary state. Protocol headers and retransmissions remain outside this parameter-payload analysis. Third, the

structured uplink exposes prompt updates, prototype summaries, counts, and reliability fields to the coordinator. Section 3.5 characterizes this message-level threat surface, derives the small-count prototype sensitivity, and establishes secure-summation and sensitivity-calibrated interfaces for the aggregation rule.

6. Conclusions

FedTIP formulates the temporal prompt adapter as the federated trainable object and uses coverage-aware joint prompt–prototype aggregation to govern both prompt deltas and class-anchor geometry. On Wikipedia, Reddit, and MOOC, it has the highest reported mean AUC-ROC in the nine same-access cells and a measured cumulative uplink of 0.40 MiB per task. The message-level analysis further characterizes prototype exposure and shows how FedTIP's additive aggregation interface composes with secure summation and sensitivity-calibrated prototype release. Together, these results establish the practical value of prompt-level federation for chronologically ordered few-shot support.

Author Contributions: Conceptualization, X.W. and X.Z.; methodology, X.W.; software, X.W.; validation, X.W.; investigation, X.W.; data curation, X.W.; visualization, X.W.; writing–original draft preparation, X.W.; writing–review and editing, X.Z.; supervision, X.Z.; project administration, X.Z.; funding acquisition, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2020YFB2103803.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Wikipedia, Reddit, and MOOC are public benchmark datasets distributed by their original providers. The reproducibility materials include chronological split manifests, user-level client assignments, task-support allocations, client-indexed validation/test attribution views, materialized support-stage negative selections, and measured communication and prototype-support summaries for the three datasets. These materials are available from the corresponding author upon reasonable request.

Acknowledgments: We wish to acknowledge the anonymous referees who gave valuable suggestions to improve the work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kumar, S.; Zhang, X.; Leskovec, J. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1269–1278. [CrossRef]
2. Trivedi, R.; Farajtabar, M.; Biswal, P.; Zha, H. DyRep: Learning Representations over Dynamic Graphs. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
3. Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; Achan, K. Inductive Representation Learning on Temporal Graphs. In Proceedings of the International Conference on Learning Representations, Virtual, 26 April–1 May 2020.
4. Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; Bronstein, M. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *arXiv* **2020**, arXiv:2006.10637. [CrossRef]
5. Kazemi, S.M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; Poupart, P. Representation Learning for Dynamic Graphs: A Survey. *J. Mach. Learn. Res.* **2020**, *21*, 2648–2720.
6. Poursafaei, F.; Huang, S.; Pelrine, K.; Rabbany, R. Towards Better Evaluation for Dynamic Link Prediction. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2022; Volume 35, pp. 32928–32941. [CrossRef]
7. Huang, S.; Poursafaei, F.; Danovitch, J.; Fey, M.; Hu, W.; Rossi, E.; Leskovec, J.; Bronstein, M.; Rabusseau, G.; Rabbany, R. Temporal Graph Benchmark for Machine Learning on Temporal Graphs. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2023; Volume 36, pp. 2056–2073. [CrossRef]

8. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*; PMLR: Cambridge, MA, USA, 2017; Volume 54, pp. 1273–1282.
9. Fu, X.; Zhang, B.; Dong, Y.; Chen, C.; Li, J. Federated Graph Machine Learning: A Survey of Concepts, Techniques, and Applications. *SIGKDD Explor. Newsl.* **2022**, *24*, 32–47. [CrossRef]
10. Liu, R.; Xing, P.; Deng, Z.; Li, A.; Guan, C.; Yu, H. Federated Graph Neural Networks: Overview, Techniques, and Challenges. *IEEE Trans. Neural Netw. Learn. Syst.* **2025**, *36*, 4279–4295. [CrossRef]
11. Huang, W.; Yang, Y.; Chen, M.; Liu, C.; Feng, C.; Poor, H.V. Wireless Network Optimization for Federated Learning with Model Compression in Hybrid VLC/RF Systems. *Entropy* **2021**, *23*, 1413. [CrossRef]
12. Li, M.; He, X.; Chen, J. Federated Collaborative Learning with Sparse Gradients for Heterogeneous Data on Resource-Constrained Devices. *Entropy* **2024**, *26*, 1099. [CrossRef]
13. Song, W.; Yan, M.; Li, X.; Han, L. Bidirectional Decoupled Distillation for Heterogeneous Federated Learning. *Entropy* **2024**, *26*, 762. [CrossRef]
14. Lester, B.; Al-Rfou, R.; Constant, N. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 3045–3059. [CrossRef]
15. Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.S.; Leskovec, J. Strategies for Pre-training Graph Neural Networks. In Proceedings of the International Conference on Learning Representations, Virtual, 26–30 April 2020.
16. Tian, S.; Wu, R.; Shi, L.; Zhu, L.; Xiong, T. Self-Supervised Representation Learning on Dynamic Graphs. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management, Virtual, 1–5 November 2021; pp. 1814–1823. [CrossRef]
17. Bei, Y.; Xu, H.; Zhou, S.; Chi, H.; Wang, H.; Zhang, M.; Li, Z.; Bu, J. CPDG: A Contrastive Pre-Training Method for Dynamic Graph Neural Networks. In Proceedings of the 40th IEEE International Conference on Data Engineering, Utrecht, The Netherlands, 13–16 May 2024; pp. 1199–1212. [CrossRef]
18. Liu, Z.; Yu, X.; Fang, Y.; Zhang, X. GraphPrompt: Unifying Pre-Training and Downstream Tasks for Graph Neural Networks. In Proceedings of the ACM Web Conference 2023, Austin, TX, USA, 30 April–4 May 2023; pp. 417–428. [CrossRef]
19. Fang, T.; Zhang, Y.; Yang, Y.; Wang, C.; Chen, L. Universal Prompt Tuning for Graph Neural Networks. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2023; Volume 36, pp. 52464–52489. [CrossRef]
20. Sun, X.; Cheng, H.; Li, J.; Liu, B.; Guan, J. All in One: Multi-Task Prompting for Graph Neural Networks. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6–10 August 2023; pp. 2120–2131. [CrossRef]
21. Chen, X.; Zhang, S.; Xiong, Y.; Wu, X.; Zhang, J.; Sun, X.; Zhang, Y.; Zhao, F.; Kang, Y. Prompt Learning on Temporal Interaction Graphs. *arXiv* **2024**, arXiv:2402.06326. [CrossRef]
22. Yu, X.; Liu, Z.; Zhang, X.; Fang, Y. Node-Time Conditional Prompt Learning in Dynamic Graphs. In Proceedings of the International Conference on Learning Representations, Singapore, 24–28 April 2025; Volume 2025, pp. 97811–97830.
23. Wu, C.; Wu, F.; Cao, Y.; Lyu, L.; Huang, Y.; Xie, X. FedGNN: Federated Graph Neural Network for Privacy-Preserving Recommendation. In Proceedings of the 2021 ICML Workshop on Federated Learning for User Privacy and Data Confidentiality, Virtual, 24 July 2021.
24. Zhang, K.; Yang, C.; Li, X.; Sun, L.; Yiu, S.M. Subgraph Federated Learning with Missing Neighbor Generation. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2021; Volume 34, pp. 6671–6682.
25. Yao, Y.; Jin, W.; Ravi, S.; Joe-Wong, C. FedGCN: Convergence-Communication Tradeoffs in Federated Training of Graph Convolutional Networks. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2023; Volume 36, pp. 79748–79760. [CrossRef]
26. Jiang, M.; Jung, T.; Karl, R.; Zhao, T. Federated Dynamic Graph Neural Networks with Secure Aggregation for Video-Based Distributed Surveillance. *ACM Trans. Intell. Syst. Technol.* **2022**, *13*, 56. [CrossRef]
27. Lou, G.; Liu, Y.; Zhang, T.; Zheng, X. STFL: A Temporal-Spatial Federated Learning Framework for Graph Neural Networks. *arXiv* **2021**, arXiv:2111.06750. [CrossRef]
28. Xie, Z.; Likun, L.; Chen, X.; Yu, H.; Huang, Q. FedDGL: Federated Dynamic Graph Learning for Temporal Evolution and Data Heterogeneity. In *Proceedings of the 16th Asian Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2025; Volume 260, pp. 463–478.
29. Yao, Y.; Fan, X.; Rossi, R.A.; Kim, S.; Zhao, H.; Yu, T.; Joe-Wong, C. Federated Link Prediction on Dynamic Graphs. In Proceedings of the NeurIPS 2025 Workshop on New Perspectives in Graph Machine Learning, San Diego, CA, USA, 7 December 2025.

30. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
31. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
32. You, J.; Du, T.; Leskovec, J. ROLAND: Graph Learning Framework for Dynamic Graphs. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 2358–2366. [CrossRef]
33. Wen, Z.; Fang, Y. TREND: Temporal Event and Node Dynamics for Graph Representation Learning. In Proceedings of the ACM Web Conference 2022, Virtual, 25–29 April 2022; pp. 1159–1169. [CrossRef]
34. Wang, Y.; Chang, Y.Y.; Liu, Y.; Leskovec, J.; Li, P. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In Proceedings of the International Conference on Learning Representations, Virtual, 3–8 May 2021.
35. Sun, X.; Zhang, J.; Wu, X.; Cheng, H.; Xiong, Y.; Li, J. Graph Prompt Learning: A Comprehensive Survey and Beyond. *arXiv* **2023**, arXiv:2311.16534. [CrossRef]
36. Long, Q.; Yan, Y.; Zhang, P.; Fang, C.; Cui, W.; Ning, Z.; Xiao, M.; Cao, N.; Luo, X.; Xu, L.; et al. Towards Graph Prompt Learning: A Survey and Beyond. *arXiv* **2024**, arXiv:2408.14520. [CrossRef]
37. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated Optimization in Heterogeneous Networks. In Proceedings of the Machine Learning and Systems, Austin, TX, USA, 2–4 March 2020; Volume 2, pp. 429–450.
38. Karimireddy, S.P.; Kale, S.; Mohri, M.; Reddi, S.J.; Stich, S.U.; Suresh, A.T. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *Proceedings of the 37th International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2020; Volume 119, pp. 5132–5143.
39. Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; Poor, H.V. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 7611–7623.
40. Acar, D.A.E.; Zhao, Y.; Navarro, R.M.; Mattina, M.; Whatmough, P.N.; Saligrama, V. Federated Learning Based on Dynamic Regularization. In Proceedings of the International Conference on Learning Representations, Virtual, 3–8 May 2021.
41. Li, X.; Jiang, M.; Zhang, X.; Kamp, M.; Dou, Q. FedBN: Federated Learning on Non-IID Features via Local Batch Normalization. In Proceedings of the International Conference on Learning Representations, Virtual, 3–8 May 2021.
42. Tan, Y.; Long, G.; Liu, L.; Zhou, T.; Lu, Q.; Jiang, J.; Zhang, C. FedProto: Federated Prototype Learning across Heterogeneous Clients. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 8432–8440. [CrossRef]
43. Park, H.; Yun, S. FedHide: Federated Learning by Hiding in the Neighbors. In *Proceedings of the Computer Vision—ECCV 2024*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2025; Volume 15128, pp. 405–422. [CrossRef]
44. Wahdany, D.; Jagielski, M.; Dziedzic, A.; Boenisch, F. Differentially Private Prototypes for Imbalanced Transfer Learning. *Proc. AAAI Conf. Artif. Intell.* **2025**, *39*, 20991–20999. [CrossRef]
45. Li, T.; Hu, S.; Beirami, A.; Smith, V. Ditto: Fair and Robust Federated Learning through Personalization. In *Proceedings of the 38th International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2021; Volume 139, pp. 6357–6368.
46. Li, X.; Wu, Z.; Zhang, W.; Zhu, Y.; Li, R.H.; Wang, G. FedGTA: Topology-aware Averaging for Federated Graph Learning. *Proc. VLDB Endow.* **2023**, *17*, 41–50. [CrossRef]
47. He, C.; Balasubramanian, K.; Ceyani, E.; Yang, C.; Xie, H.; Sun, L.; He, L.; Yang, L.; Yu, P.S.; Rong, Y.; et al. FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks. *arXiv* **2021**, arXiv:2104.07145. [CrossRef]
48. Yao, Y.; Li, Y.; Fan, X.; Li, J.; Liu, K.; Jin, W.; Yang, Y.; Ravi, S.; Yu, P.S.; Joe-Wong, C. FedGraph: A Research Library and Benchmark for Federated Graph Learning. *arXiv* **2024**, arXiv:2410.06340. [CrossRef]
49. Lin, Y.; Wei, Y.; Chen, D.; Li, Y.; Erkan, U.; Toktas, A.; Gao, S.; Zhang, Y. Cryptanalysis and Improvement of a Video Cryptosystem via Chaos and S-Box. *ACM Trans. Multimed. Comput. Commun. Appl.* **2026**, *22*, 173. [CrossRef]
50. Bonawitz, K.A.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2017; pp. 1175–1191. [CrossRef]
51. Dwork, C.; Roth, A. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* **2014**, *9*, 211–407. [CrossRef]
52. Snell, J.; Swersky, K.; Zemel, R.S. Prototypical Networks for Few-Shot Learning. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30, pp. 4077–4087.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.